Mathieu BESSON

Comment garantir la continuité de transmission des connaissances et des savoir-faire au sein d'une TPE dans le domaine de l'informatique afin de limiter la perte de savoirs critiques ?

Mémoire professionnel

MBA Développeur full-stack, MyDigitalSchool Rennes

Référent professionnel : Christophe HIRON, christophe.hiron@blue2i.fr, 02 23 37 35 69

Maitre de mémoire : SCHUHMACHER Paul, paul.schuhmacher.ext@eduservices.org

Référent pédagogique : Nadia FÉREC, nadia.ferec@aftec.fr

Date de rendu : Juin 2024

Abstract

Dans le contexte dynamique d'une petite entreprise du domaine de l'informatique, la gestion et la transmission des connaissances et des savoirs-faire sont cruciales, notamment pour limiter les pertes de savoirs et maintenir la productivité des salariés. Les TPE et PME[1] sont fréquemment confrontées à des défis importants dans ce domaine : outils dépréciés, connaissances dispersées ou non documentées ou encore gestion de projets inefficaces.

Pour répondre à ces enjeux majeurs dans l'évolution d'une entreprise, ce mémoire vise à identifier les axes d'amélioration et les solutions à mettre en place pour répondre à ces défis. En analysant les besoins spécifiques de l'entreprise, les différentes recherches réalisées dans le domaine et en s'appuyant sur des retours d'expérience, ce mémoire vise à proposer des solutions pour organiser, partager et maintenir de manière durable les connaissances et les savoirs-faire dans une entreprise du secteur de la tech.

La *SARL*[2] *blue2i* est une agence spécialisée dans le web et dans le développement sur-mesure, basée à Châteaubourg près de Rennes. Elle compte actuellement 7 collaborateurs répartis en trois pôles : création de visuels, de maquettes et de supports de communication, développement de sites vitrine avec le CMS[3] *Drupal*, et création d'applications sur-mesure avec le framework *Symfony*[4]. Le cas pratique de ce mémoire est centré sur la problématique rencontrée par cette entreprise, spécialisée dans le développement informatique, et plus précisément sur la gestion de sa documentation technique.

Au cours de mon alternance chez *blue2i*, j'ai travaillé sur plusieurs missions visant notamment à améliorer la gestion des connaissances et la documentation technique de l'entreprise. J'ai organisé et normalisé la documentation existante, mis en place un outil d'agrégation de la documentation et des procédures, et proposé des solutions pour limiter les points de friction à la rédaction et à la mise en commun des connaissances. Ces missions ont permis de répondre aux besoins de l'entreprise et de justifier l'intérêt du sujet de ce mémoire sur la thématique de l'importance de la gestion et de la transmission efficace des connaissances dans une petite entreprise du secteur de l'informatique.

[1] TPE : Très petite entreprise, PME : Petite ou moyenne entreprise
[2] SARL : Une société à responsabilité limitée est une forme de société commerciale qui permet aux associés de limiter leur responsabilité au montant de leurs apports.
[3] CMS (Content Management System) : Système de gestion de contenu, logiciel grâce auquel il est possible de créer, de gérer et de modifier facilement un site web. Drupal est l'un de ces outils, www.drupal.org
[4] Framework Symfony : Framework (ou cadre de travail) du langage de programmation PHP destiné aux développeurs pour créer des sites ou applications web complexes, symfony.com/

Remerciements

Je tiens tout d'abord à remercier mon entreprise d'accueil, qui m'a permis de réaliser ce mémoire dans de bonnes conditions. Je suis particulièrement reconnaissant envers mon CTO et responsable, Christophe HIRON, qui a su me faire confiance et me donner du temps pour travailler sur un projet intéressant et formateur autant sur l'aspect technique, conception et réalisation que sur l'aspect stratégique.

Je tiens également à remercier mon maître de mémoire Paul SCHUHMACHER, qui m'a accompagné et guidé pour l'élaboration de ce mémoire. Ses conseils et son expertise m'ont été d'une grande aide pour structurer mon plan et rédiger mon mémoire dans les meilleures conditions.

Je remercie mes parents, Gabriel et Anne-Sophie, qui ont toujours été là pour moi, pour me soutenir durant toute la durée de mes études. Je remercie également Pauline et Coralie, pour leurs encouragements et leur soutien continu qui a été pour moi d'une grande aide.

Enfin, je souhaite adresser mes remerciements à l'ensemble des personnes qui ont contribué, de près ou de loin, à la réussite de ce projet. Qu'il s'agisse de mes collègues, qui ont pris le temps de répondre à mes questions et de m'apporter leur aide, ou des différents intervenants extérieurs qui ont accepté de partager leur expérience et leur savoir-faire avec moi.

Table des matières

Glossaire

API : *Interface de programmation d'application*, permettant à des logiciels de communiquer entre eux.
Auto-hébergé : Se dit d'un logiciel ou d'un site web hébergé sur son propre serveur.
Build : Processus de construction de contenu à partir d'une source.
CI/CD : *Intégration continue et livraison continue*, pratique de développement logiciel intégrant des tests automatisés et une livraison continue des mises à jour.
CMS Drupal : *Content Management System*, Système de gestion de contenu open-source écrit en PHP.
Doc block : Bloc de doc dans au dessus d'une fonction ou d'une classe dans le code source d'un logiciel.
EPUB : Format de fichier ouvert pour les livres électroniques.
Framework : Structure logiciel pré-établis structurant le développement de logiciels.
Framework Symfony : Framework PHP open-source pour le développement web.
HTML : Langage de balisage standard pour la création de pages web.
IDE : Environnement de développement intégré, un logiciel pour le développement de logiciels.
Langage de markup : Langage de balisage utilisé pour le formatage de documents.
Markdown : Langage de balisage léger pour la mise en forme de documents.
Normes PSR : *PHP Standard Recommendation*, ensemble de normes de codage pour les projets PHP.
Onboarding : Processus d'intégration de nouveaux employés.
Open-source : Logiciel dont le code source est accessible et modifiable par tous, sous certaines contraintes.
OS : *Operating System,* système d'exploitation, le logiciel de base d'un ordinateur.
Parsable : Se dit d'un texte ou d'un code pouvant être analysé par un programme.
PHP : Langage de programmation open-source utilisé pour le développement web.
Pipeline : Série de tâches automatisées dans un processus de développement logiciel.
POC : *Proof of Concept*, Preuve de concept ou démonstration de faisabilité
Pull Request (PR) : Demande d'intégration de modifications de code dans un projet.
Python : Langage de programmation open-source utilisé pour de nombreuses applications.
RH : *Ressources humaines*, le département responsable de la gestion du personnel.
SARL : *Société à responsabilité limitée*, une forme d'entreprise.
TPE/PME : *Petites et moyennes entreprises* (0 - 10 et 10 - 250 salariés).
Turn-over : Rotation du personnel dans une entreprise.
UI/UX/DX: *User Interface, User Experience et Développeur Experience*, les aspects d'un logiciel liés à l'utilisateur.
UML : *Unified Modeling Language*, modélisation standard pour la conception logiciels.
Wiki : Outil de visualisation de documentation technique.
WYSIWYG : *What you see is what you get*, éditeur de texte affichant le résultat final.
XML : Langage de balisage utilisé pour stocker et transporter des données.

Introduction

Justification de l'intérêt du sujet

La gestion des connaissances est un enjeu important pour les entreprises, particulièrement dans un secteur dynamique comme l'informatique. Cette partie de l'introduction permet d'examiner rapidement son historique et sa pertinence actuelle, les défis spécifiques rencontrés par les entreprises informatiques, et le cas particulier d'une très petite entreprise (TPE) comme *blue2i*.

Historique et sujet d'actualité

La gestion des connaissances est un enjeu crucial pour les entreprises de tous les secteurs, y compris celui de l'informatique. Dans un environnement en constante évolution, la capacité à gérer efficacement les savoirs est un facteur clé de succès pour les entreprises. En effet, la gestion des connaissances permet de capitaliser sur les savoir-faire et les compétences des collaborateurs. De plus, un pilotage efficace de la documentation permet de concentrer les ressources pour favoriser la réel plus value des collaborateurs : l'innovation et la créativité. Les objectifs attendus par les entreprises sont l'amélioration de la productivité et de la qualité des produits et services, et le renforcement de la compétitivité de l'entreprise sur son marché.

Enjeux dans le secteur de l'informatique

Aujourd'hui, selon le *Blog du modérateur*[5], un collaborateur tech change de poste tous les "2 à 3 ans en moyenne", avec un turn-over estimé "entre 25% et 30%", par an. Ainsi, la gestion des connaissances reste un enjeu majeur pour les entreprises, en particulier dans un secteur de l'informatique volatile. Dans cet environnement, les entreprises font face à de nouveaux enjeux tels que la perte de connaissance et de savoir au départ de salariés cadres, l'onboarding lent des nouveaux arrivants ou encore la nécessité de partager les connaissances entre les différents pôles d'activité. Une gestion efficace de ces éléments permet à terme de fluidifier le transfert de compétences dans le cadre des onboarding mais aussi de la transmission de projets client. Une bonne gestion améliore la productivité et l'autonomie des collaborateurs et permet d'allouer des ressources plus importantes à la production de valeurs de manière compétitive.

Le cas spécifique d'une TPE

Dans le cas d'une TPE comme *blue2i*, la gestion des connaissances est un enjeu encore plus crucial. En raison de sa petite taille et de sa structure organisationnelle plus

[5] Patard Alexandra, Bilan et perspectives de l'emploi dans la tech en 2023., www.blogdumoderateur.com, 9 janvier 2023.

simple, l'entreprise est très sensible au turn-over et au renouvellement de ses collaborateurs. Sans mesure efficace, le départ de certains cadres de l'entreprise aura pour conséquence des pertes de connaissances et de savoir-faire mais aussi une baisse significative de la productivité. Dans ce contexte, en 2021, au départ de collaborateurs cadres, *blue2i* fait face à des défis importants pour gérer plus efficacement ses connaissances et limiter les pertes de savoir-faire. En conséquence, d'autres enjeux émergent pour ce type d'entreprise comme l'intégration efficace des nouveaux collaborateurs et le partage efficace de connaissances en interne.

Les défis auxquels l'entreprise est confrontée sont représentatifs des défis auxquels de nombreuses petites et moyennes entreprises font face dans le secteur de l'informatique. En identifiant les axes d'amélioration et en proposant des solutions adaptées pour répondre à ces défis, ce mémoire vise à apporter une contribution significative à la résolution de ces enjeux importants pour l'évolution du cadre de travail dans ce type d'entreprise.

Identification de la problématique en entreprise

Aujourd'hui, la problématique à laquelle est confrontée *blue2i* est liée à la gestion des connaissances et des compétences de ses collaborateurs. En effet, le départ de collaborateurs cadres entraîne une perte de connaissances et de compétences essentielles pour l'entreprise. La perte de savoir-faire en interne rend l'intégration de nouveaux collaborateurs plus complexe et complique la gestion des projets longue durée, mais aussi des projets émergeants.

La problématique pratique de l'entreprise qui guide ce mémoire est donc la suivante : Le départ de collaborateurs cadres entraîne la perte de valeurs/connaissances essentielles, une complexité accrue d'intégration des nouveaux collaborateurs et des difficultés dans la gestion et l'avancement des projets.

Question de recherche

La problématique identifiée précédemment a conduit à la formulation de la question de recherche suivante :

Comment garantir la continuité de transmission des connaissances et des savoir-faire au sein d'une TPE dans le domaine de l'informatique afin de limiter la perte de savoirs critiques ?

Cette question de recherche sera le fil conducteur de ce mémoire et permettra d'orienter la réflexion vers des solutions concrètes et adaptées à la situation de l'entreprise.

Phases de raisonnement : Présentation du plan

La question de recherche étant la suivante : *"Comment garantir la continuité de transmission des connaissances et des savoir-faire au sein d'une TPE dans le domaine de l'informatique afin de limiter la perte de savoirs critiques ?"*. Le plan présenté ci-dessous permet d'apporter des éléments de réponse à cette question en proposant une démarche d'analyse structurée allant de l'analyse générale du secteur au cas spécifique de blue2i pour finir par une mise en application pratique.

La première partie de ce plan, intitulée *"Analyse, recherches et hypothèses"*, vise à comprendre les enjeux liés à la gestion de la documentation technique dans le secteur de la tech et à identifier les solutions existantes. Elle comprend une analyse des problématiques liées à la perte de connaissances, au turnover et aux coûts de maintenance de la documentation, ainsi qu'une étude des moyens et outils utilisés dans le secteur pour résoudre ces problématiques. Cette partie présente aussi les objectifs d'un outil de gestion de documentation technique et les attentes générales des entreprises du milieu.

La deuxième partie, intitulée *"Contexte de l'entreprise, conception et mise en application"*, se concentre sur la mise en œuvre de solutions concrètes pour améliorer la gestion de la documentation technique dans une entreprise spécifique. Elle comprend une analyse contextualisée de l'entreprise, une identification des enjeux et des objectifs ciblés, ainsi qu'une présentation de l'architecture, des choix techniques et de la stratégie organisationnelle retenue. Elle détaille également la méthodologie proposée pour la conception, la production et la livraison des solutions, ainsi que les premiers éléments de retour utilisateurs et d'évaluation d'efficacité de ces solutions. En fin de partie, celle-ci propose également d'éventuels ajustements et perspectives d'évolution après retour d'expérience des premières mises en application.

Ce plan permet de répondre à la question de recherche en proposant des solutions concrètes et adaptées au contexte de l'entreprise, tout en évaluant leur efficacité et leur pertinence.

Analyse, recherches et hypothèses

Pour comprendre les enjeux associés à la gestion des connaissances et des savoirs-faire dans une entreprise de la tech, il est important d'analyser son environnement. Dans ce contexte, il est essentiel de comprendre les problématiques liées à cette gestion, ainsi que d'identifier les solutions existantes pour y répondre de manière efficace. Cette partie est une analyse approfondie de l'espace des problèmes et des solutions, en s'appuyant sur des études et des exemples concrets issus du domaine de la tech. Elle présente les objectifs et les attentes générales en termes de gestion de documentation technique, mais aussi des pistes de réflexion et d'amélioration de la qualité et l'efficacité de cette gestion au sein des entreprises.

Analyse de l'espace des problèmes

Cette sous-partie à pour objectif d'analyser les problématiques importantes du secteur informatique, en particulier la gestion du turnover. En se basant sur des études récentes et des données chiffrées, cette partie analyse l'impact de la rotation des effectifs sur les entreprises du domaine. Cette analyse permettra de mieux comprendre les enjeux majeurs liés à ce phénomène, d'identifier des hypothèses de recherche et de proposer des solutions adaptées pour améliorer la gestion des connaissances et permettre une meilleure intégration des nouveaux collaborateurs.

Études dans l'environnement de la tech

Constat de la rotation des effectifs

Dans le cadre de l'étude de l'environnement du milieu de l'informatique, il est intéressant d'identifier les défis majeurs du secteur. Un enjeu particulièrement important est le taux de turnover des effectifs des entreprises de la tech. Estimé en 2011 à 15% par *MUNCI*[6], le taux de turnover est maintenant de 20% (Étude *HEC*[7]) contre une moyenne de 15% tous secteurs confondus selon *l'INSEE*[8]. Ce phénomène entraîne de nombreuses conséquences négatives sur les entreprises du milieu : la perte de connaissances techniques/métiers non documentées, un coût de formation et d'onboarding important des nouveaux collaborateurs. De plus, la multiplication des entrées et sorties de collaborateurs dans une entreprise entraîne une perte de productivité importante des ressources humaines en poste liée notamment au renouvellement et à la formation incessante des nouveaux effectifs. Dans les causes principales de ce turnover on retrouve : des salaires pas assez

[6] MUNCI (Mouvement pour une Union Nationale des Consultants en Informatique), Consultation public pour l'élaboration du plan France numérique 2020, www.economie.gouv.fr, 30 Septembre 2011
[7] HEC (Hautes Etudes Commerciales), Nouvelles perspectives pour réduire l'impact du turnover dans l'informatique, www.hec.fr
[8] Claude Picart, INSEE (Institut national de la statistique et des études économiques), Une rotation de la main d'œuvre presque quintuplée en 30 ans plus qu'un essor des formes particulières d'emploi, un profond changement de leur usage, www.insee.fr, Avril 2014

compétitifs, une sous-utilisation des compétences, des possibilités d'évolution limitées, un système de recrutement inadapté ou encore un mauvais management.

Impact du turnover

Cette rotation des collaborateurs entraîne indéniablement de nombreux impacts négatifs sur les entreprises du secteur.

Dans un premier temps, l'impact financier direct représente un enjeu majeur notamment à cause des coûts liés aux recrutements et à la formation ou encore à la perte de productivité des ressources humaines consacrées à la formation des nouveaux arrivants. De plus, les collaborateurs quittant la société emportent avec eux leurs connaissances, savoir-faire et expertises techniques non transmis aux autres collaborateurs. Selon une étude *Gallup* de 2019[9], le remplacement d'un employé peut coûter à l'entreprise entre la moitié et le double de son salaire annuel. Le coût estimé de la perte de connaissance métiers/technique suite au départ d'un développeur reste proportionnel à l'étendue de ses responsabilités au sein de l'entreprise mais aussi à la part des connaissances et savoir-faire non transmis à ses collaborateurs. L'entreprise du secteur du recrutement et de l'intérim *Randstad*[10] précise aussi qu'en plus des frais directs de renouvellement d'un salarié tel que les agences de recrutement, les frais de RH et managers ou encore les coûts de formation et d'intégration d'un nouveau salarié, on retrouve aussi bon nombre de coûts indirects. Parmi eux, la perte de temps de travail liée à la formation, la lenteur des performances initiales ou encore le coût des ressources de formation et d'intégration initiales. Ainsi, d'après un exemple de *Welcome To The Jungle*[11], média spécialiste du travail et de l'emploi, tous frais confondus (direct et indirect), un salarié embauché à 42 000€ brut annuels quittant l'entreprise après 6 mois coûterait 55 675€ à l'entreprise. Le déploiement d'outils spécifiques pour limiter les frictions et la durée d'intégration d'un nouveau salarié pourrait permettre de réduire ces coûts et faciliter l'onboarding.

Dans un second temps, cette situation a aussi un impact sur les effectifs déjà en place. On retrouve un désengagement important des employés lié au renouvellement incessant des collaborateurs, une augmentation du stress et de leur charge de travail ou encore des difficultés à garder ou à recruter les talents. En effet, face à cette situation, le désengagement des employés déjà en place peut aussi représenter un coût important de productivité pour l'entreprise. Cette perte d'efficacité est généralement liée aux multiples renouvellements et onboarding des nouveaux collaborateurs créant une fatigue et pression sur les autres salariés. En moyenne, selon l'*IBET (Indice de Bien-Être au Travail) 2020*[12], le

---

[9] Shane McFeely et Ben Wigert, Gallup (Entreprise de services de recherche sur la gestion du management, des ressources humaines et les statistiques), This Fixable Problem Costs U.S. Businesses $1 Trillion, www.gallup.com, 13 Mars 2019

[10]Randstad (Entreprise spécialisé dans l'intérim et les ressources humaines) Le coût réel du turnover en entreprise, www.randstad.fr, 17 mai 2023

[11] Charlotte Legrand, Welcome To The Jungle (Média spécialiste du travail et de l'emploi), Combien coûte vraiment un mauvais recrutement ?, www.welcometothejungle.com, 12 avr. 2021

[12] IBET (Indice de Bien-Être au Travail), Groupe APICIL, Communiqué de presse : Qualité de vie au travail : un salarié désengagé coûte 14 310€ par an à son entreprise, selon l'IBET 2020, www.groupe-apicil.com, 16 Octobre 2020

coût moyen du mal-être au travail et du désengagement par an et par salarié s'élève à 14 310 € dont 9 010 € sont des coûts maîtrisables permettant d'atteindre le niveau haut de “bonne pratique”. Ainsi, en plus d'affecter directement les entreprises sur leur productivité immédiate, la rotation des salariés entraîne manifestement des coûts secondaires importants sur la productivité des collaborateurs historiques des entreprises.

Pour finir, cet impact se fait aussi ressentir sur les clients et les prestations de service. Parmi les conséquences principales ayant une influence directe sur le client on retrouve une baisse de la qualité des services ou encore l'augmentation des délais de production.

Importance de l'onboarding et du partage efficace des connaissances

L’intégration d’un nouveau salarié dans une entreprise est un moment important autant pour l'entreprise que pour le salarié. Selon l’étude *“Onboarding Software Solutions”*[13], 22 % des départs de salariés ont lieu dans les 45 premiers jours suivant leur embauche, coûtant à l'entreprise au minimum trois fois le salaire de l'ancien collaborateur. Cela montre l'importance de la qualité de l'onboarding, qui peut avoir un impact significatif sur la rétention des employés, la productivité et la satisfaction au travail. L’onboarding doit permettre au nouveau collaborateur de comprendre les attentes et les objectifs de l'entreprise, et de disposer des outils et des connaissances nécessaires pour réussir dans son nouveau poste. Enfin, le partage efficace des connaissances et des procédures est essentiel pour garantir la continuité des savoirs-faires de l'entreprise notamment durant l’intégration de nouveaux salariés.

Comme identifié précédemment, le turnover est un phénomène fréquent dans les entreprises, en particulier chez les profils techniques dans le domaine de l’informatique. Ces derniers ont une grande valeur en termes de production et de connaissances, et leur départ peut avoir un impact important sur l'activité et la productivité de l'entreprise. Il est donc essentiel de mettre en place des outils pour conserver les savoirs en interne et faciliter le transfert de connaissances entre les salariés. Cela permet de limiter les conséquences négatives du turnover, mais aussi de faciliter l'intégration des nouveaux collaborateurs en leur donnant accès au savoir-faire internes partagés par les autres collaborateurs.

Le partage efficace des connaissances est un enjeu crucial pour les entreprises du domaine, car il permet de réduire les pertes de temporelles et financières liées à la recherche d'informations et à la duplication de tâches. Selon le *Rapport Panopto*[14] sur la connaissance et la productivité du lieu de travail, les travailleurs du tertiaire perdent en moyenne 5,3 heures par semaine à attendre des informations essentielles de la part de leurs collègues ou à travailler pour recréer les connaissances institutionnelles existantes.
Cette même étude révèle que 42 % des connaissances institutionnelles sont individuelles donc non partagées dans l’entreprise, expliquant cette perte de productivité importante.

---

[13] Katherine Jones, Ph.D, Onboarding Software Solutions 2014, Bersin by Deloitte (Deloitte Consulting LLP), 2014
[14] Rapport Panopto, Le partage inefficace des connaissances coûte 47 millions de dollars par an aux grandes entreprises, www.panopto.com, 31 juillet 2020

Le coût de la perte de productivité liée au non échange des informations est considérable. Toujours selon Panopto, une entreprise de 3 000 employés perd en moyenne 8 millions de dollars par an à cause de ce phénomène. Il est donc essentiel pour les entreprises de mettre en place des outils et des processus permettant de faciliter le partage des connaissances entre les collaborateurs.

Le coût de l'implémentation et de la maintenance de ces outils et processus peut sembler important, mais il est largement compensé par les gains de productivité réalisés. En effet, la réduction du temps alloué à la recherche continue d'informations permet aux collaborateurs de se concentrer sur des tâches à plus forte valeur ajoutée, augmentant la productivité globale de l'entreprise. De plus, en facilitant l'intégration des nouveaux collaborateurs grâce à un accès plus facile aux connaissances institutionnelles, l'entreprise limite aussi les ressources liées à la formation et à l'accompagnement de ces derniers.

Stratégies de gestion de la connaissance dans le secteur

La documentation des conventions et normes par le code

La première documentation d'un projet vient du code source lui-même. En suivant des bonnes pratiques de nommage, de structure et de commentaires, le code peut devenir auto-documenté et plus facile à comprendre pour les autres développeurs. En se basant sur le code existant et en suivant les normes et les conventions établies, les développeurs peuvent produire du code de qualité et maintenable tout en contribuant à la maintenabilité des applications produites.

Cette idée est confirmée par une étude intitulée *"Usage and usefulness of technical software documentation: An industrial case study"*[15], réalisée en partie par Vahid Garousi, professeur de génie logiciel à l'université de Queen's à Belfast. Cette étude, menée au sein de *NovAtel Inc.*, a pour but de comprendre l'utilisation et l'utilité de la documentation technique pendant le développement et la maintenance de logiciels. Elle a analysé 55 documents et 1630 révisions, et a sondé 25 membres du personnel expérimentés. Les résultats montrent que la documentation technique est peu consultée pour la maintenance et que le code source est la source d'information privilégiée. De plus, il n'y a pas de différence significative entre l'utilisation des différents types de documents.

Cette étude montre qu'au fil du temps et de la maintenance d'un logiciel, le code source est de plus en plus considéré comme la source de vérité, car la documentation technique n'est pas systématiquement mise à jour et peut donc devenir obsolète. Ainsi, cela révèle l'importance de la documentation du code source lui-même. En se basant sur le code et la documentation existants dans le code source tout en suivant les normes et les conventions établies, les développeurs peuvent continuer de produire du code de qualité et maintenable, et contribuer à la maintenabilité des logiciels. Ainsi, plus la documentation reste proche du code source, plus celle-ci est susceptible d'être mise à jour en même temps

[15] Garousi G., Garousi-Yusifoɬlu V., Ruhe G., Zhi J., Moussavi M., Smith B., Usage and usefulness of technical software documentation: An industrial case study., Information and Software Technology, 2015

que celui-ci, permettant ainsi de limiter la friction à la mise à jour et d'accroître la durabilité de celle-ci dans le temps.

En développement logiciel, il est nécessaire de suivre des bonnes pratiques communes pour garantir la qualité, la maintenabilité et la durabilité des applications. De nombreux ouvrages de référence, tels que *"Clean Code"*[16] de *Robert C. Martin* ou *"Software Craftsmanship"*[17] de *Pete McBreen*, présentent des principes et des techniques pour produire du code propre, bien structuré et facile à comprendre. Ces écrits posent une structure établie des bonnes pratiques à adopter dans la production de code source quelque soit le langage ou la technologie.

Dans le langage de programmation *PHP* qui est utilisé dans chez blue2i, les normes de programmation *PSR (PHP Standard Recommendations)*[18] sont largement adoptées et reconnues comme des références en termes de bonnes pratiques. Ces normes, élaborées par le groupe de travail *PHP-FIG (PHP Framework Interop Group)*[19], permettent d'établir des guides de référence sur des aspects comme la structure et le nommage des fichiers, la gestion des dépendances, la gestion des erreurs et des exceptions, la sécurité ou encore les normes de documentation du code source.

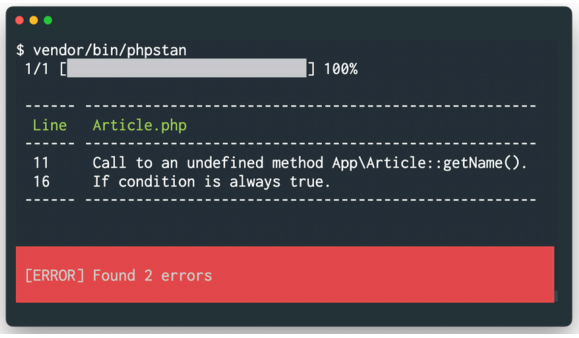
En suivant les normes *PSR*, les développeurs *PHP* peuvent garantir un niveau de qualité et de prévisibilité élevé du code source en facilitant la collaboration et le partage de code avec d'autres développeurs. De plus, ces standards servent de base pour le développement d'outils permettant ainsi d'accélérer les flux de développement, tels que les linters (détecteurs statiques d'anomalies syntaxiques ou conventionnelles), les interpréteurs de documentation de doc block, ou encore la documentation auto-générée. Dans l'univers *PHP*, il est possible d'intégrer des outils comme *PHPStan*[20] permettant de contrôler la qualité du code et le respect des conventions et bonnes pratiques. L'intégration de cet outil d'analyse de code dans une pipeline de déploiement continue *CI/CD* permet de s'assurer de la qualité du code. La détection efficace de ces anomalies par la pipeline de développement permet de réduire les risques de bug et d'anomalies en production ainsi que d'élever le niveau de qualité et de fiabilité des applications déployées.

---

[16] Martin Robert C., Clean Code: A Handbook of Agile Software Craftsmanship, Prentice Hall, 2008
Principes clean code adaptés à PHP : www.github.com/piotrplenik/clean-code-php
[17] McBreen Pete, Software Craftsmanship: The New Imperative, Addison-Wesley Professional, 2001
[18] Les normes PSR (PHP Standards Recommendations) sont un ensemble de normes et de recommandations pour la codification et la conception de logiciels en PHP, www.php-fig.org/psr
[19] Le PHP-FIG est un groupe de travail créé pour promouvoir l'interopérabilité entre les projets PHP en élaborant des normes. www.php-fig.org
[20] PHPStan est un outil d'analyse statique de code PHP, conçu pour détecter les erreurs de développement et améliorer la qualité du code. www.phpstan.org

```php
/**
 * This is a Summary.
 *
 * This is a Description. It may span multiple lines
 * or contain `code` examples using the _Markdown_ markup
 * language.
 *
 * @see Markdown
 *
 * @param int        $parameter1 A parameter description.
 * @param \Exception $e          Another parameter description.
 *
 * @\Doctrine\Orm\Mapper\Entity()
 *
 * @return string
 */
function anything(int $parameter1, Exception $e)
{
    ...
}
```

Exemple de doc block PHP respectant les standards PSR



```
$ vendor/bin/phpstan
 1/1 [                            ] 100%

------ ------------------------------------------------------
 Line   Article.php
------ ------------------------------------------------------
 11     Call to an undefined method App\Article::getName().
 16     If condition is always true.
------ ------------------------------------------------------

[ERROR] Found 2 errors
```

Exemple d'une sortie d'analyse statique de code par PHPStan

Les différents types de documentation

Dans son article *"Software documentation"*[21]*, Ian Sommerville* définit trois types de documentation pour les logiciels : la documentation du processus, la documentation du système et la documentation de l'utilisateur.

La documentation du processus (Process Documentation) regroupe l'ensemble des documents liés à la conception du produit, à la maintenance, à la qualité, à l'organisation projet. Elle permet de comprendre les étapes et les décisions prises lors de la création du logiciel. Ce type de document regroupe les normes et conventions projet, les procédures de support et de maintenance logiciel ou encore les spécifications fonctionnelles du projet.

La documentation du produit (System Documentation) vise à produire des documents décrivant le système conçu. Plus formellement, elle décrit comment fonctionne le logiciel du point de vue des développeurs et comment le faire évoluer. Ce type de documentation est principalement destiné aux développeurs et aux équipes techniques travaillant sur le logiciel. Dans ce type de document on retrouve les schémas d'architecture et les spécifications techniques du logiciel.

Enfin, la documentation de l'utilisateur (User documentation) est totalement orientée vers les utilisateurs finaux du logiciel. Elle est adaptée aux compétences des lecteurs et aux besoins de chaque utilisateur pour qu'ils puissent utiliser le logiciel de manière efficace. Pour ce type de document, il est possible de rédiger plusieurs documents sur la même thématique en fonction du lecteur cible et de son besoin spécifique.

[21] Sommerville Ian, Software Documentation, Lancaster University

Comparaison des outils existants

Il existe aujourd'hui une multitude d'outils de gestion de la documentation, qu'ils soient propriétaires ou open-source. Parmi les plus populaires, on retrouve *Confluence*[22], *Gitbook*[23] et *Notion*[24]. Le tableau récapitulatif ci-dessous compare les différentes fonctionnalités principales des outils du marché. Ce dernier permettra de déterminer quel outil est le plus adapté aux besoins spécifiques de l'entreprise en matière de gestion de la documentation.

Fonctionnalités	Gitbook	Confluence	Notion	Wiki.js[25]	MkDocs[26]	Docusaurus[27]
Création et édition de contenu directement depuis l'interface	✅	✅	✅	✅	❌	❌
Intégrations externes	Slack	Jira, Trello	Slack, Github			
Versions et historique des modifications	✅	✅	✅	✅ (git)	✅ (git)	✅ (git)
Permissions et autorisations d'accès	✅	✅	✅	✅ (avec Gitea[28])	✅ (avec Gitea)	✅ (avec Gitea)
Snippets et templates de docs	✅	✅	✅	❌	❌	❌
Personnalisation apparence	✅	❌	❌	✅	✅	✅
Recherche contenu	✅	✅	✅	✅	✅	✅

[22] Confluence : logiciel de collaboration et de gestion de contenu, conçu pour aider les équipes à créer, organiser et partager des connaissances et édité par la société Altassian. www.atlassian.com/fr/software/confluence
[23] GitBook est une plateforme de création et de publication de documentation technique, basée sur Git et Markdown. www.gitbook.com
[24] Notion est un logiciel de productivité et de gestion de contenu, conçu pour aider les utilisateurs à organiser et à partager des informations et des idées. www.notion.so
[25] Wiki.js est une plateforme de wiki open-source, conçue pour aider les utilisateurs à créer et à gérer des wikis collaboratifs. www.js.wiki
[26] MkDocs est un générateur de site statique pour la création de documentation, basé sur Markdown. squidfunk.github.io/mkdocs-material
[27] Docusaurus est un générateur de site statique, basé sur Markdown. www.docusaurus.io
[28] Gitea : Plateforme open-source de gestion de version de code collaborative source basé sur le git, c'est un outil concurrent de Github ou Gitlab, about.gitea.com/

Gestion différents types de contenu (image, texte, vidéos, fichiers, tableaux, listes)	✅	✅	✅	✅	✅	✅
Import git (markdown)	Github, Gitlab	❌	❌	✅	✅	✅
Pricing	6.7$ /user /month	4.89$ /user /month	Free plan	Free	Free	Free
Ouvert à la customisation ou à la mise à jour	❌	❌	❌	❌	✅	✅
Type	Propriétaire	Propriétaire	Propriétaire	Open source	Open source	Open source

Des captures d'écran de présentation des différents outils sont disponibles en annexe 1.

Pour une entreprise détenant en interne les compétences de mise en place et de maintenance, il peut être intéressant d'utiliser un outil open-source. En effet, un outil open-source présente plusieurs avantages par rapport aux outils propriétaires pour la gestion de la documentation. Tout d'abord, il permet une customisation et des développements sur mesure facilités grâce à son architecture modulaire, ses multiples extensions et à sa communauté active de développeurs. Les éventuelles fonctionnalités manquantes peuvent largement être compensées par des développements sur-mesure parallèles. L'entreprise garde la main sur le déploiement et les données de sa documentation, ce qui permet une administration et une gestion plus souple et plus sécurisée. De plus, ce type d'outil permet une customisation complète et intègre souvent bien la syntaxe markdown en y ajoutant de nombreux modules complémentaires permettant de répondre aux besoins spécifiques de l'entreprise. Enfin, par l'aspect open-source de ces outils, l'entreprise est à l'abri des éventuels changements de tarification comme sur les outils propriétaires comme *Notion* ou *Confluence* qui peuvent mettre à jour leur formules d'un jour à l'autre.

MkDocs : Retour d'expérience technique d'un outil open-source

Dans son mémoire de fin d'étude *"Piloting markdown-bases documentation"*[29], *Eero Mäkiranta* présente une étude de cas sur le déploiement d'un outil de documentation basé sur le langage Markdown dans une entreprise de conception de systèmes sur puce (*SoC*). Son objectif est de résoudre les problèmes de documentation éparpillée, de manque de lien entre la documentation et le code source, de gestion d'accès insuffisante et de lourdeur des logiciels déjà en place. Dans son mémoire, *Mäkiranta* propose la mise en place d'un portail

[29] Mäkiranta Eero, *Piloting markdown-bases documentation*, Tempere University, 2023

de documentation basé sur *Markdown*. Cet outil a été mis en place et testé dans le cadre d'un projet pilote pour la documentation de projets de petite envergure *SoC*.

Les résultats de cette étude de cas montrent que l'utilisation d'un outil de documentation basé sur le langage *Markdown* peut être bénéfique dans un contexte technique, en particulier pour la documentation de projets complexes comme les *SoC*. Par sa syntaxe simple et efficace, le langage *Markdown* encourage les développeurs à mettre à jour leur documentation en même temps que le code source de leur projet. Le stockage des documentations par l'intermédiaire d'un gestionnaire de versions comme *git* offre aussi de nombreux avantages tels que l'historisation des versions ou encore le contrôle et la gestion des mises à jour par pull-request. L'utilisation du *Markdown* permet aussi de multiplier les cibles de production de la documentation comme vers un site web statique, en *PDF ou* vers d'autres formats. Au-delà de rendre la documentation plus accessible, l'utilisation de ce langage de balisage rend durable les documentations grâce à sa simplicité et à la standardisation du langage le rendant indépendant des plateformes cibles de build.

En conclusion, cette étude de cas fournit un retour d'expérience précieux sur l'aménagement d'un outil de documentation basé sur le langage *Markdown* dans une entreprise de conception de *SoC*. Les résultats positifs de cette initiative montrent que cette approche peut être bénéfique dans des contextes techniques similaires.

Objectifs et synthèse des attentes générales

Lors de la mise en œuvre d'une solution de documentation globale pour une entreprise, plusieurs aspects doivent être pris en compte. Tout d'abord, la solution choisie doit permettre une création rapide et efficace de la documentation, de manière à réduire le temps de rédaction et la friction à la mise à jour. La documentation produite doit également être générique pour pouvoir s'adapter à différents types de projets et contraintes techniques (schémas, images, listes, niveaux de titres…), tout en restant facile à utiliser pour les rédacteurs réguliers.

Un autre aspect important est la concentration sur le contenu plutôt que sur la forme. La solution de documentation doit permettre de visualiser de façon lisible et attrayante le contenu tout en garantissant la simplicité de mise à jour et la clarté du contenu. Il est essentiel que la documentation soit synthétique, concise et à jour pour assurer une compréhension globale du projet et éviter les mauvaises interprétations.

De plus, la solution de documentation doit offrir un contrôle adéquat en termes d'autorisations de modification. Les utilisateurs doivent avoir les permissions appropriées pour accéder, modifier et partager les documents. Cependant, les propositions de modification doivent être soumises à un processus d'approbation pour garantir la qualité et la cohérence de la documentation.

La centralisation des documents est également un élément clé dans le choix d'une solution. Elle doit permettre la gestion de documents provenant de différentes sources et

destinés à différents publics cibles, en garantissant une cohérence et une harmonisation de la documentation finale produite. Ainsi, quelle que soit l'origine de la doc (projet, dépôt dédié, librairie…), la solution doit permettre de limiter la friction à la mise à jour, en offrant des fonctionnalités telles que la synchronisation automatique avec le code source. Le format de production doit être durable dans le temps et permettre à terme de construire la documentation finale pour différents formats de cibles. Dans cette optique, la documentation doit être lisible autant à partir du code source que depuis l'extérieur et permettre la multidiffusion sur plusieurs cibles et formats potentiels.

En conclusion, la mise en place d'une solution de documentation globale pour une entreprise est un choix stratégique qui doit prendre en compte les objectifs et attentes de celle-ci. La solution de rédaction choisie doit être efficace, générique, axée sur le contenu plutôt que sur la forme, tout en étant attrayante et lisible. Il doit également être possible de contrôler efficacement les modifications grâce à un système d'autorisations et de validation. Enfin, la solution doit permettre de limiter la friction à la mise à jour de la documentation, en offrant une syntaxe efficace et une synchronisation efficiente avec l'outil de visualisation final. Ces éléments clés permettent ainsi de définir les outils les plus efficaces pour répondre aux besoins d'une entreprise.

Analyse de l'espace des solutions

Après avoir analysé l'espace des problèmes et les enjeux liés à la problématique dans le secteur de l'informatique, cette partie s'intéresse aux différentes solutions et outils existants dans le domaine de la gestion efficace des connaissances en entreprise. Nous examinerons différents aspects comme le code source comme première source de documentation, la gestion des contenus avec git, les formats de rédaction légers à travers les langages de markups, les outils de build de markup open-source, mais aussi les pratiques d'organisation et de rédaction technique pour garantir la durabilité des connaissances.

Le code source : Première source de documentation

En développement logiciels, le code source est la principale référence pour comprendre le fonctionnement d'un système. Il est essentiel de considérer le code source comme la première source de documentation. En respectant les normes de codage telles que les conventions de nommage *PSR*, les règles syntaxiques et en utilisant un analyseur statique de code *PHP*, le code devient auto-documenté et plus facile à comprendre. Plus concrètement, l'installation d'outils comme *PHPStan* permet de faire respecter les normes et conventions *PHP* en rendant le contrôle de code systématique et efficace. Une fois établies, ces règles rendent le code auto-documenté et guide l'intégration des nouveaux développeurs arrivant sur un projet. *PHPStan* proposant un niveau de règle progressif, cela facilite l'intégration de cet outil dans de nouveaux projets.

Pour garantir le respect des conventions et bonnes pratiques, il est possible d'utiliser des hooks pré-commit et pré-merge de pull-request (PR) afin de vérifier le code avec

PHPStan avant de le soumettre à la revue de code ou de le fusionner avec la branche principale.

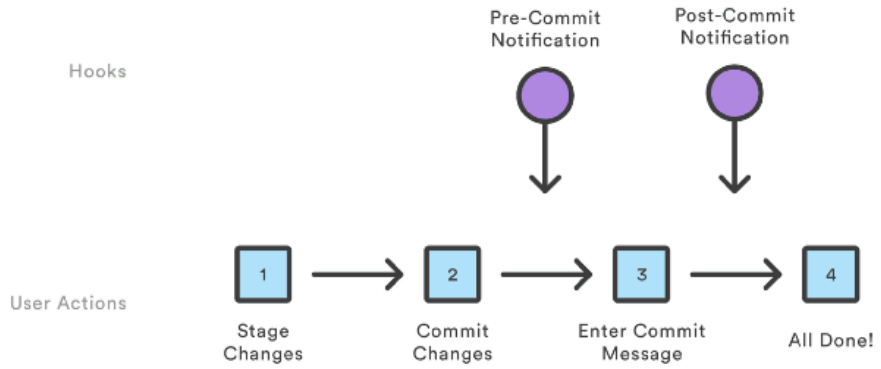
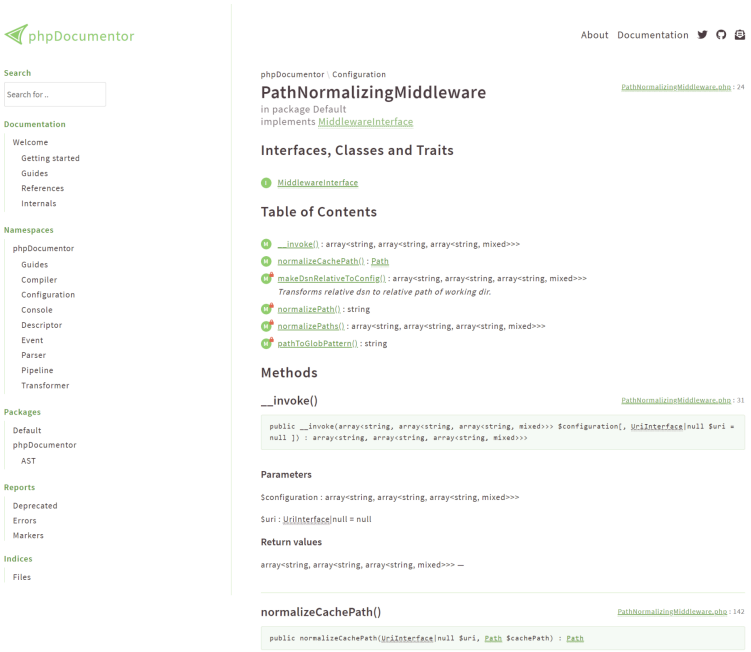


Schéma explicatif des événements de hooks git[30]

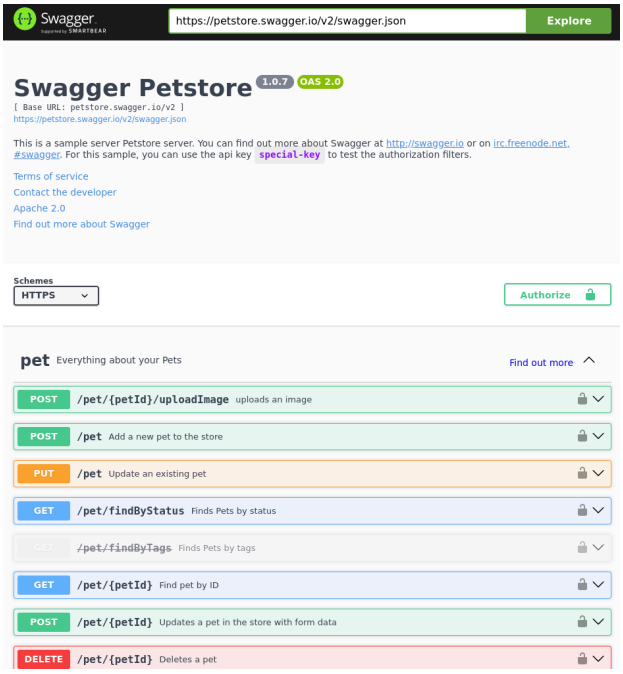
Il est possible d'extraire la documentation à partir du code source, en utilisant des outils tels que *PHPDocumentor*[31] pour générer de la documentation dans différents formats (*PDF*, web, etc.). À partir du code source, *PHPDocumentor* analyse et extrait les “doc block” de manière à générer une documentation statique permettant de visualiser les informations sur les classes, fonctions et attributs. Cet outil permet de générer automatiquement la documentation à partir du code source, évitant aux développeurs la rédaction manuelle de celle-ci. Associée à un outil d’automatisation, cette documentation est toujours synchronisée avec le code source garantissant sa cohérence et augmentant la qualité de la base de code.



Exemple de génération statique de documentation de code par PHPDocumentor[32]

[30] Kanani Nirav, Github Pre-commit Hook Setup In Ruby On Rails for maintaining coding standards and productive, 2022, dev.to/kanani_nirav/github-pre-commit-hook-setup-in-ruby-on-rails-12m3
[31] phpDocumentor est un outil de génération de documentation pour le langage PHP, basé sur les commentaires de documentation dans le code source. www.phpdoc.org
[32] phpDocumentor, Because code and documentation are meant to be together, www.phpdoc.org

Les documentations des endpoint d'*API* peuvent également être générées automatiquement à partir du code source en utilisant des normes d'annotation de style doctrine[33] et des outils tels que *Swagger* et *OpenAPI*[34]. La documentation étant stockée dans le code source lui-même, les frictions à la mise à jour de la documentation de l'*API* sont fortement réduites.



Exemple de Swagger auto-généré depuis le code source[35]

```php
<?php

use OpenApi\Annotations as OA;

class MyController {

  /**
   * @OA\Get(
   *   path="/products",
   *   summary="list products",
   *   @OA\Response(
   *     response=200,
   *     description="A list with products"
   *   ),
   *   @OA\Response(
   *     response="default",
   *     description="an ""unexpected"" error"
   *   )
   * )
   */
    public function resource() {
        // ...
    }
}
```

Annotations OpenAPI au format Doctrine (normes PSR)

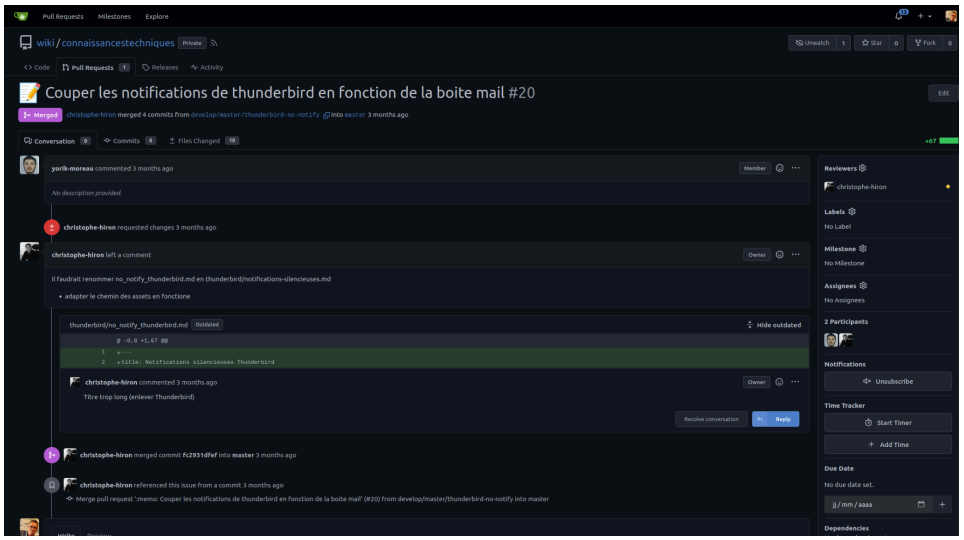
Pour finir, l'utilisation d'un éditeur de code interprétant la documentation de "doc block", tel que *Visual Studio Code* ou encore *PhpStorm*[36], facilite également la compréhension du code et de la documentation. Ces intégrations, par les *IDE,* offrent une visualisation instantanée des types des variables, des commentaires ou encore des exemples d'utilisation, facilitant la complétion du code, le débogage et l'utilisation du code source.

Gestion des contenus avec git

La gestion de la documentation avec *git*[37] offre de nombreux avantages pour les équipes de développement. *Git* permet de gérer la documentation facilement en parallèle du code sans besoin d'outils supplémentaire. L'utilisation d'un gestionnaire de version permet de garder une trace de l'historique des modifications, la gestion de branches et la gestion

[33] Les normes d'annotation de style Doctrine sont des conventions pour l'utilisation des annotations dans le langage PHP, spécifiquement pour le framework ORM Doctrine. zircote.github.io/swagger-php/guide/annotations.html
[34] Swagger est un outil pour la génération de documentation d'api tandis que OpenAPI est une norme de spécifications pour la description d'API RESTful. swagger.io, www.openapis.org
[35] Capture d'écran d'un exemple de Swagger : petstore.swagger.io/
[36] Visual Studio Code est un éditeur de code, tandis que PhpStorm est un environnement de développement intégré (IDE), code.visualstudio.com www.jetbrains.com/fr-fr/phpstorm
[37] Git est un système de contrôle de version décentralisé, utilisé pour la gestion de code source.

éventuels des conflits de version. Couplé à l'utilisation de pull request sur un gestionnaire de version collaboratif comme *GitHub* ou *Gitlab*[38], les revues de code permettent un contrôle efficace de la documentation produite et déployée en production.



Exemple de pull-request Gitea permettant la revue de code par un collaborateur

Ensuite, les outils comme *Github* ou *Gitea* offrent de nombreuses fonctionnalités pouvant faciliter le déploiement automatique de cette documentation. Ainsi, on retrouve dans ces outils des webhooks permettant de notifier des outils externes des éventuels événements ayant lieu sur un dépôt de code. La détection de ces événements offre une réelle valeur ajoutée permettant le déclenchement d'un processus automatisé et personnalisé de mise à jour en temps réel de la documentation finale produite sous forme de wiki (web), de *PDF* ou *EPUB* (livre numérique)... Cet aspect limite les frictions à la mise à jour des documents et encourage les équipes à maintenir la documentation à jour, au plus près du code, tout en sachant que celle-ci sera déployée de façon automatique sur les cibles spécifiques.

L'utilisation d'un gestionnaire de version distant pour la gestion des contenus documentaires permet aussi une structuration personnalisée des dépôts pouvant être séparés par thématiques/catégories (procédures, connaissances…), par type de cible ou encore par autorisation. Ce type d'outil facilite l'implémentation de développements sur mesure garantissant une interconnexion efficace entre les différents outils.

[38] *GitHub*, *Gitlab et Gitea sont des plateformes de gestion de code source et de collaboration basé sur git,* about.gitea.com github.com about.gitlab.com

Formats de rédactions légers et fonctionnels : les langages de markups

Les langages de markup légers

Il existe de multiples langages de markup légers permettant une rédaction efficace et intuitive. Ces langages simples sont en général utilisés pour formater du texte en vue d'une publication web ou sur d'autres supports. Ces langages aux fonctionnalités limitées permettent la mise en évidence des titres, des listes, des tableaux, des liens hypertextes, des images, etc. sans avoir à écrire de code *HTML* plus verbeux. Il existe de nombreux langages de markup légers, chacun ayant ses spécificités et ses avantages. Parmi les plus connus, on peut citer :

- *Wikitext*[39], utilisé par *Wikipedia* et ses dérivés, et sa version standardisée *Creole*
- *Asciidoc*[40], un langage markup texte dédié à la rédaction de contenus techniques
- *Textile*[41], un langage de markup simple et intuitif
- *reStructuredText*[42] (rst, reST), utilisé notamment dans la documentation *Python* avec Sphinx

Exemple de contenu identique formaté avec les différents langages de markups légers

Le Markdown

Le langage de markup léger le plus répandu et qui offre le plus de compatibilité avec d'autres outils est le *Markdown*. Créé en 2004 par *John Gruber et Aaron Swartz*, il est aujourd'hui utilisé par de nombreux éditeurs de texte, outils de blogs, forums, et même par des outils de gestion de versions comme *GitHub* ou *GitLab*. Son objectif principal est de permettre une mise en forme de texte simple et intuitive, tout en restant lisible et facile à écrire.

[39] Wikitext : fr.wikipedia.org/wiki/Wikitexte
[40] Asciidoc : https://asciidoc.org
[41] Textile : textile-lang.com
[42] reStructuredText : www.sphinx-doc.org

Exemple de contenu formaté avec Markdown

Dans la publication de la première version de *Markdown*[43], *John Gruber* présente l'objectif de *Markdown* : "l'objectif primordial de la syntaxe de formatage de *Markdown* est de la rendre aussi lisible que possible. L'idée est qu'un document formaté en *Markdown* doit pouvoir être publié tel quel, en texte brut, sans donner l'impression d'avoir été marqué par des balises ou des instructions de formatage."

Le *Markdown* a été initialement conçu comme un outil de conversion de texte en *HTML* pour les rédacteurs souhaitant écrire pour le web de manière simple et efficace. Par la suite, il a rapidement gagné en popularité dans différentes situations, allant du développement logiciel à la rédaction scientifique en passant par le blogging. Par sa simplicité et son efficacité, le markdown est rapidement devenu un choix populaire par rapport aux langages de balisage lourd comme *HTML* ou *XML*. Il est désormais omniprésent sur le web avec des sites tels que *Reddit*[44] et *GitHub* prenant en charge ce format, ainsi que de nombreux éditeurs *WYSIWYG*.

Exemple de l'intégration de markdown dans un README.md Github en annexe 2.

Avantages et inconvénients du Markdown

Les avantages du *Markdown* par rapport à d'autres outils sont nombreux :

- Il est simple, intuitif et facile à apprendre et à maîtriser. Les symboles utilisés pour la mise en forme sont identifiables facilement et permettent de se concentrer sur le contenu plutôt que sur la forme.
- C'est un langage de balisage léger et efficace. Il est lisible par les humains et facilement parsable et analysable par des robots ou des processus automatisés.
- Il permet aussi de rédiger du contenu rapidement et facilement, sans avoir à utiliser des outils de formatage complexes. Contrairement aux outils de rédaction classiques, tels que *Microsoft Word* ou *Google Docs*, le formatage est effectué directement dans le contenu, ce qui évite les pertes de temps liées à la manipulation de l'interface utilisateurs.

[43] Gruber John, Markdown, www.daringfireball.net/projects/markdown, 2004
[44] Reddit est un site web de partage de contenu et de discussion, organisé en communautés thématiques.

- Il est portable et peut être utilisé sur tous les *OS*, étant stocké sous forme de fichier texte brut, cela en fait un choix idéal pour la gestion de versions et la collaboration.
- Le *Mardown* est compatible avec de nombreux outils, tels que *GitHub*, *WordPress*[45] ou encore les forums de discussion. Il permet de publier du contenu vers le web, le *PDF* ou encore *EPUB (livre numérique)*, à partir d'une seule et même source, ce qui facilite la gestion de contenu vers des cibles multicanal.
- Ce format est durable et à l'épreuve du temps, en cas de changement de solution de visualisation, le contenu n'a pas à être modifié pour évoluer vers de nouveaux outils.

Malgré bon nombre d'avantages il subsiste quelques inconvénients à l'usage de Markdown, notamment au niveau de la standardisation du langage. Le manque de standardisation de sa syntaxe peut entraîner des confusions et des erreurs d'interprétation. Il n'existe pas de spécification officielle de *Markdown* et l'auteur ne veut pas étendre la syntaxe pour garder l'aspect simple du langage.

Au fil du temps, la communauté a fini par inventer ses propres implémentations indépendantes appelées "flavors"[46]. Ces implémentations permettent l'ajout de fonctionnalités supplémentaires non disponibles à l'origine, tels que les alertes, la coloration syntaxique des extraits de code… Parmi ces nouvelles implémentations, on retrouve le *GitHub Flavored Markdown (GFM)*[47] : la version du markdown créée par *Github*, *Python Markdown*[48] pour des outils *Python* ou encore *Markdoc*[49] une implémentation open-source orientée création de contenu développée par *Stripe* pour la création de leur documentation structurée et complexe. Ces différentes versions peuvent entraîner des problèmes de conversion d'un système à l'autre à travers des syntaxes incompatibles. Pour résoudre cette situation, une initiative non-officielle de normalisation des différentes "flavors" de *Markdown* nommée *CommonMark*[50] a vu le jour. Cette initiative vise à établir une spécification syntaxique standard et sans ambiguïté pour *Markdown*. L'intégration de cette norme permettra à terme de réduire les incompatibilités gênantes entre les différentes implémentations indépendantes.

Il existe bon nombre d'exemples de documentations de projets open source rédigés à l'aide de Markdown, qui permettent de construire par la suite des documentations en ligne : *MDN (Mozilla Developer Network)*, *AdonisJS* (framework *node.js* back-end), *Ionic* (framework cross-plateforme), *React Native* (framework mobile cross-plateforme) ou encore *Jest* (framework de test pour *node.js*). Ainsi, *Markdown* s'impose aujourd'hui comme le choix de référence pour la rédaction de documentations de projets en raison de sa simplicité, de sa portabilité et de sa compatibilité avec de nombreux outils.

Exemples de documentations construites à partir du format Markdown en annexe 3.

[45] WordPress est un système de gestion de contenu (CMS), utilisé pour la création et la gestion de sites web et de blogs.
[46] Liste des différentes "flavors" : www.github.com/commonmark/commonmark-spec/wiki/markdown-flavors
[47] Format Markdown de GitHub (GFM) : github.github.com/gfm
[48] Python Markdown : python-markdown.github.io
[49] Markdoc : markdoc.dev
[50] CommonMark : spec.commonmark.org

Outils de build de markup open source

Une fois le langage de markup identifié il peut être nécessaire d'utiliser un ou plusieurs outils de build permettant de créer des documents cibles avec des formats spécifiques.

Pour ce type de besoins, l'utilisation d'outils open-source présente de nombreux avantages. En effet, dans la plupart des cas, ces outils sont gratuits et auto-hébergés permettant de garder un contrôle sur les données et une gestion autonome de l'outil. De plus, ils bénéficient généralement de nouvelles fonctionnalités régulières et de corrections de bug par les différents contributeurs. L'utilisation d'outils open-source offre une grande flexibilité en termes d'organisation, de personnalisation et de développement parallèles sur mesure, permettant de répondre au mieux aux besoins d'une entreprise.

Il existe deux types d'outils de build open source pour la création de documentation à partir de langages de markup légers tels que *Markdown* : les outils de build globaux et les outils de build spécifiques.

Le premier type : les outils globaux, prennent en charge de nombreux langages de markup et peuvent convertir les fichiers sources en différents formats cibles tels que *HTML*, *EPUB*, *XML*, *PDF*, etc. On retrouve dans ce type d'outils *Pandoc*[51] ou encore *DITA Open Toolkit*[52] *(DITA-OT)*.

Le deuxième type est celui des outils plus spécialisés qui se concentrent sur une cible spécifique, dans notre cas le web, pour la création de wikis ou de sites statiques. Ces outils peuvent offrir des fonctionnalités supplémentaires telles que la génération de menus de navigation, la recherche en texte intégral, la gestion des versions ou même une UI/UX moderne et épurée. Parmi eux, on retrouve *MkDocs Material*, *Jekyll*[53], *Docusaurus* ou encore *Wiki.js*.

Les choix d'outils de build dépendent des besoins et des exigences spécifiques de chaque entreprise ou projet. En effet, pour une documentation au format livre numérique on privilégiera des outils comme Pandoc, alors que pour la génération de wiki on se tournera vers *MkDocs* ou *Docusaurus*. Les outils globaux peuvent offrir une plus grande flexibilité en termes de formats de sortie, tandis que les outils plus spécialisés peuvent offrir des fonctionnalités supplémentaires pour une cible spécifique.

[51] Pandoc : www.pandoc.org
[52] DITA Open Toolkit : www.dita-ot.org
[53] Jekyll : jekyllrb.com

Pratiques d'organisation et de rédaction technique

Cadre de travail, approches et outils

Cette partie se concentre sur la formation à la pratique de l'écriture de documentation technique. L'objectif est d'identifier les bonnes pratiques en termes d'organisation et de conception de documentation et en termes de rédaction.

Cadre de travail

Dans l'optique d'établir un cadre de travail structuré et clair pour la rédaction de documentation technique, il peut être intéressant d'utiliser un framework tel que *Diataxis*[54]. Le framework *Diataxis* est une approche de qualité pour la rédaction de documentation technique. Développé par *Daniele Procida*, Director of Engineering à Canonical, ce cadre de travail présente 4 modes de documentation :

- **Tutoriels** (Tutorials) : Guide à travers les tâches pour accomplir un objectif de manière didactique, sans ou avec peu de connaissances préalables requises
- **Guides pratiques** (How-to guides) : Instructions à la réalisation d'une tâche spécifique de manière concise et directe, les procédures en sont un parfait exemple.
- **Explications** (Explanations) : Vulgarisation et explication de concepts, processus, choix de conception ou fonctionnalités.
- **Références** (Reference) : Manuel de référence détaillé, complet et factuel sur un sujet donné.

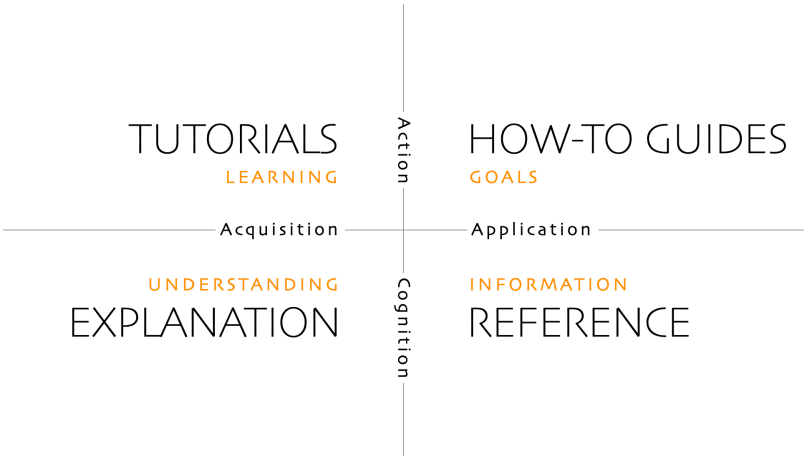


Schéma explicatif des différents types de documentations dans le cadre Diataxis[55]

Le framework *Diataxis* conseille des bonnes pratiques à appliquer pour chacun des modes de documentation :

- **Tutoriels** :
  - Se concentrer sur une tâche spécifique avec un objectif clair et précis.

[54] Procida Daniele, Diátaxis, diataxis.fr
[55] Schéma explicatif du cadre Diataxis : diataxis.fr/

    - Diviser les tâches en étapes faciles à suivre, les instructions doivent être claires et concises.
    - Utiliser des exemples concrets et des captures d'écrans pour faciliter la compréhension.
    - Lier des ressources complémentaires.
    - Éviter les termes techniques, les jargons spécialisés.

- **Guides pratiques** :
    - Définir un besoin ou un problème spécifique à résoudre.
    - Fournir des instructions détaillées étapes par étapes pour résoudre le problème avec des exemples concrets et des captures d'écran.
    - Se concentrer sur l'essentiel mais envisager les différents cas de figures.
    - Fournir des liens vers des ressources supplémentaires pour approfondir les connaissances.

- **Explications** :
    - Définir le concept ou le choix de conception à expliquer.
    - Expliquer de manière simple les termes ou concepts techniques complexes.
    - Fournir des exemples concrets pour illustrer les concepts et les processus.
    - Utiliser des diagrammes et des schémas pour faciliter la compréhension.

- **Références** :
    - Fournir des informations détaillées, complètes et factuelles sur un sujet spécifique.
    - Utiliser une structure claire et cohérente pour organiser les informations.
    - Fournir des liens vers des ressources supplémentaires pour approfondir les connaissances.
    - Utiliser des tableaux et des listes pour faciliter la recherche d'informations.

L'utilisation du framework *Diataxis* permet de mieux identifier les besoins et le niveau de connaissance du public cible. L'objectif est de définir la structure, le langage et le style à adopter en fonction du type de cible et du niveau d'information à transmettre. Ce cadre de travail est un précieux outil pour le rédacteur, lui fournissant une ligne directrice dans son processus d'élaboration d'une documentation de qualité.

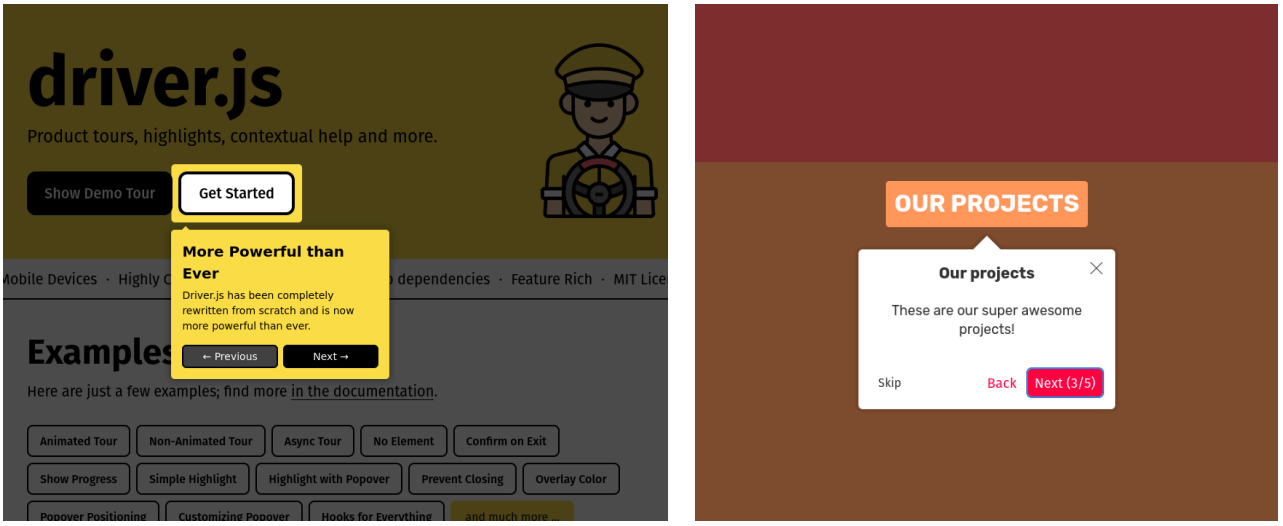
Documentation projet

En termes de conception et de documentation projet, une approche efficace est le *"Documentation-Driven Design" (conception basée sur la documentation)*. Cette approche consiste à élaborer la documentation avant le développement du projet. Ce modèle permet de communiquer efficacement la solution proposée à toute une équipe. Par exemple, dans un projet avec une architecture complexe mettant en jeu plusieurs *API* il peut être utile de définir les spécifications de ces *API* comprenant des exemples de réponses et de requêtes ou encore des schémas d'architecture. L'établissement de ces spécifications, en amont du projet, permet de clarifier les attentes et les besoins du projet avant même le démarrage du développement du projet. De plus, la création de schémas d'architecture, des spécifications

et des interactions entre les différentes ressources permet une compréhension claire et commune des fonctionnalités, exigences et contraintes du projet. Enfin, la conception basée sur la documentation permet de réduire les coûts de changements. En effet, plus les anomalies et malentendus seront identifiés tôt, moins les efforts de développement à investir seront significatifs. Il sera, en effet, toujours moins coûteux de mettre à jour une documentation que le code source d'un projet.

Documentation utilisateur

Concernant la documentation utilisateur, il est toujours plus efficace de privilégier un guide utilisateur embarqué à l'application. Ces nouveaux types de documentations embarquées sont directement importées du monde du jeu vidéo. Ils offrent l'avantage de former efficacement le nouvel utilisateur par la pratique à travers des étapes successives menant à la découverte d'une ou plusieurs fonctionnalités. L'établissement d'un tel outil peut rapidement permettre de diminuer les coûts en support client pour un logiciel. Par exemple, l'entreprise *Sophos*[56], spécialiste dans le milieu de la cybersécurité, a constaté une réduction de plus de 12 000 tickets de support annuels grâce à l'utilisation de guides interactifs intégrés dans leur console d'administration web. Cet impact sur le service support d'une entreprise de cette taille, démontre l'efficacité des guides interactifs in-app en termes de formation des nouveaux utilisateurs sur une plateforme. Il existe de nombreux outils permettant l'intégration facile de guides d'onboarding in-app pour les applications web, parmi eux on retrouve des outils comme *Driver.js*[57] ou *React Joyride*[58].



Exemples d'utilisation de guide utilisateur in-app avec Driver.js et React Joyride

[56] Olmstead Levi, 9 Types of In-App Training to Enable Users (+Examples), Whatfix, 2024, whatfix.com/blog/in-app-training
[57] Driver.js : driverjs.com
[58] React Joyride : react-joyride.com

Bonnes pratiques de rédaction

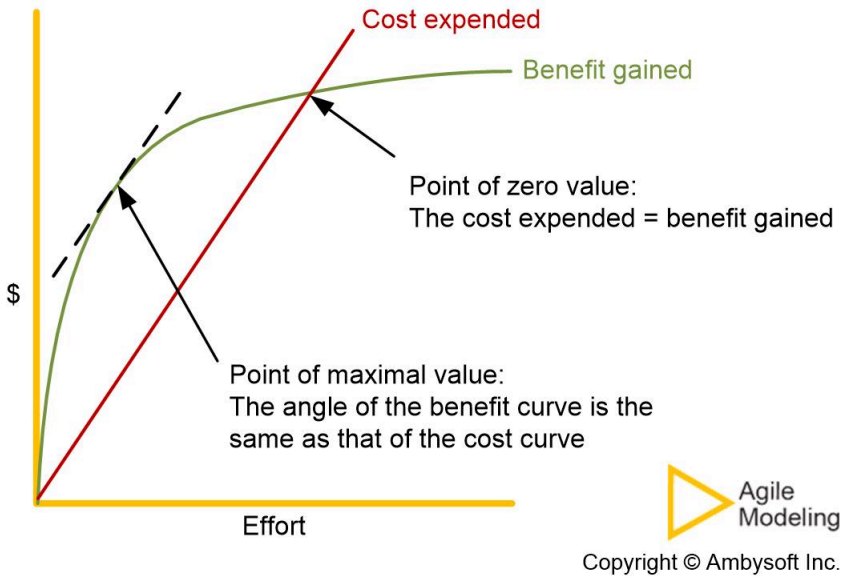
Lors de la rédaction de documentation technique, il est important de suivre certaines bonnes pratiques pour garantir la clarté, la précision et l'efficacité des informations présentées.

“Less is more” et “just barely good enough”

Le principe populaire *"Less is more"* s'applique aussi au monde de la documentation technique. En effet, il vaut mieux privilégier une documentation concise et précise plutôt qu'une documentation trop longue et redondante qui risque de ne pas être lue. Il est également important de distinguer la rédaction de documentation dans un contexte de projet "traditionnel" et dans un contexte dit “agile”. Dans un contexte traditionnel, la documentation peut être utilisée comme un outil de suivi de projet et de décision, conduisant à terme à une surcharge de documentaire inutile. Dans un contexte agile, la documentation est généralement plus légère et plus ciblée sur quelques documents majeurs et durables comme des schémas d'architecture projet, des conventions ou encore des spécifications techniques. Le *Manifest Agile*[59] préconise d'ailleurs "un logiciel fonctionnel plutôt qu'une documentation exhaustive". Il faut ici comprendre une documentation légère, ciblée sur le besoin utilisateur et facile à maintenir, apportant de la valeur tout en garantissant sa durabilité et son efficacité.

Ainsi, il est important de ne pas tout documenter, mais plutôt de se concentrer sur les concepts durables et non éphémères. Les schémas d'architecture et les conventions de développement sont des exemples de concepts durables qui méritent d'être documentés. En revanche, le suivi de projet n'a pas besoin d'être documenté car rendant, à terme, la gestion de la documentation difficile et la documentation inopérante. D'après un adage tiré des réflexions autour de l’agilité, il est important de décrire *"juste ce qu'il est bon de savoir"* dans la documentation. Il faut adopter une approche *"tout juste suffisante" ("just barely good enough"* [60]*en anglais)* visant à fournir suffisamment d'informations pour répondre aux besoins de l'utilisateur, mais pas plus. Cette approche permet de gérer efficacement la production et la maintenance des documentations en gardant en ligne directrice le rapport coût/bénéfice/effort associé. Le diagramme suivant décrit les coûts et les bénéfices associés à la création d'une documentation. Il montre que pour maximiser la valeur, il est important de trouver le point où le coût d'ajouter plus d'efforts dépasse la valeur ajoutée. Ce point est appelé *"Just Barely Good Enough" (JBGE)*. Cependant, toute proportion gardée, il est important de noter que les courbes des coûts et des bénéfices peuvent être affectées par différents facteurs et peuvent donc ne pas être aussi lisses que dans le diagramme théorique.

[59] Beck, Kent, et al. , Manifesto for Agile Software Development, 2001, agilemanifesto.org
[60] "just barely good enough", agilemodeling.com/essays/barelygoodenough.htm



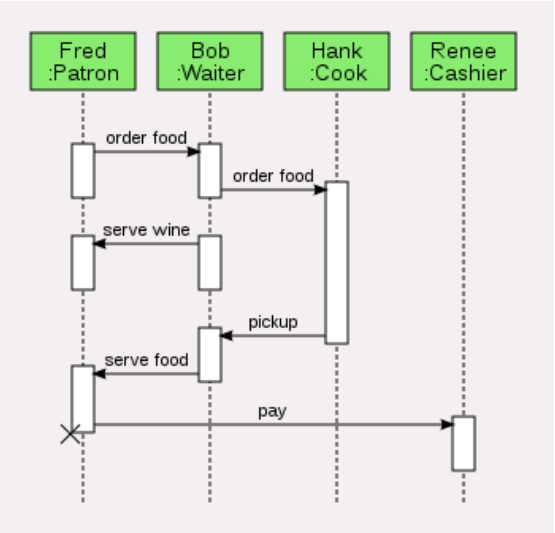
Graphique explicatif du concept de "Just Barely Good Enough" (JBGE)[61]

La schématisation : un outil de communication efficace

Dans le cas de la description d'un système complexe mettant en interaction de multiples ressources, il peut être utile de mettre en place des outils visuels spécifiques pour mieux comprendre ce système. Dans ces cas de figure, l'utilisation de diagrammes ou de schémas explicatifs peut être une solution efficace.
Par exemple, les schémas *UML (Unified Modeling Language)* sont des représentations graphiques standardisées utilisées pour modéliser un système. Il existe plusieurs types de schémas *UML*, mais certains sont plus utilisés que d'autres :

- *Diagramme de classe* : Modélise la structure statique d'un système de classes interagissant entre elles par l'intermédiaire de relations.
- *Diagramme de base de données* : Modélise la structure des tables d'une base de données, leur relations, leurs champs et types de champs et les spécificités de chaque table.
- *Diagramme cas d'utilisation* : Modélise la façon dont les différents acteurs (utilisateurs) interagissent avec un système et leurs possibilités.
- *Diagramme d'activité* : Modélise les étapes réalisées dans un cas d'utilisation, ce type de diagramme permet d'exprimer le comportement dynamique et conditionnel d'un système.
- *Diagramme de séquence* : Modélise les différents événements, scénarios et intéractions entre éléments.

[61] Just Barely Good Enough (JBGE) Artifacts: An Agile Core Practice, agilemodeling.com/essays/barelygoodenough.htm



Exemple de diagramme UML de séquence[62]

Les diagrammes *UML* sont des outils puissants et flexibles pour la conception et la documentation de systèmes complexes. Les différents types de diagrammes *UML* offrent chacun des avantages et des utilisations spécifiques pour modéliser et communiquer les différents aspects d'un système. Leur utilisation facilite la communication et la collaboration autour d'un projet. Les diagrammes *UML* sont des outils puissants mais malgré tout ils ne conviennent pas à tous les besoins et à toutes les situations. Dans certains cas, des schémas plus simples et plus rapides à créer peuvent être plus efficaces pour communiquer et faire comprendre des concepts. Les diagrammes *UML* doivent donc être utilisés avec discernement, en fonction des objectifs, des contraintes et des préférences de l'équipe de projet.

Recommandations des leaders du marché

Les leaders du marché de la tech présentent eux aussi leurs recommandations quant au processus de rédaction de document technique. À travers les ressources de *"Write The Docs"*[63], une communauté regroupée autour des meilleures pratiques de création de documentation logicielle ou encore *"Technical Writing Courses"* [64]de *Google*, il est possible d'identifier les bonnes pratiques à adopter en termes de rédaction technique. À travers ces référentiels on retrouve les recommandations suivantes :

- **Identifier et cibler le public** et utilisateur de la doc : utilisateurs finaux, développeurs, administrateurs... L'objectif est d'adapter le contenu et la forme de manière à répondre aux besoins spécifiques de cette cible.

[62] Diagramme de séquence, Exemple de diagramme de séquence d'un restaurant, fr.wikipedia.org/wiki/Diagramme_de_s%C3%A9quence
[63] Write the Docs, www.writethedocs.org
[64] Google, Technical Writing Courses (Google), developers.google.com/tech-writing

- **Utiliser des exemples pratiques** : exemples de code, de diagrammes, de captures d'écran ou de vidéos, de cas concrets pour illustrer les concepts et faciliter la compréhension.

- **Traiter la documentation comme du code source** : outil de versionnement, suivi de l'historique, gestion des conflits de modification. Cette approche permet de mettre en place une collaboration efficace et de garantir l'évolution de la doc en parallèle du code source.

- **Maintenir à jour la doc** : à l'image des modifications apportées au produit, la documentation doit aussi être mise à jour pour éviter d'être rendue obsolète. Une documentation incohérente est plus nuisible que son absence totale.

- **Feedback et contributions externes** : encourager les retours d'expériences et les contributions des utilisateurs finaux aide à l'identification des lacunes et à apporter des améliorations.

Suivre ces recommandations, partagées par des acteurs d'expérience comme *Google* ou *Write The Docs,* permet aux rédacteurs techniques de mettre en place une documentation plus efficace et ciblée pour leurs utilisateurs finaux.

En conclusion de cette première partie, il est clair que la gestion de la documentation technique est un enjeu crucial pour les entreprises du secteur de la tech. Les défis liés à la perte de connaissances, au turnover et aux coûts de maintenance de la documentation sont réels et nécessitent des solutions concrètes et adaptées.

Comme vu précédemment les objectifs d'un outil de gestion de documentation technique sont multiples : faciliter le partage de connaissances et la collaboration, réduire les coûts de maintenance et d'onboarding ou encore améliorer la qualité et l'efficacité de la documentation parfois inexistante ou dispersée.

Les approches théoriques et outils techniques utilisés dans le secteur pour résoudre ces problématiques sont nombreuses et variées allant de simples méthodes organisationnelles, à des solutions de gestion de contenu en ligne, tout en passant par les systèmes de gestion de versions de code plus classiques.

Cependant, il est important de noter que l'implémentation d'outils de gestion de documentation technique ne peut à elle seule résoudre l'ensemble des défis liés à la gestion de la connaissance dans une entreprise. Elle doit être accompagnée d'une réflexion plus globale sur les pratiques et les processus de l'entreprise concernant la transmission des connaissances et des savoirs-faire.

Dans la deuxième partie de ce mémoire, nous nous concentrerons sur la mise en œuvre de solutions concrètes pour améliorer la gestion de la documentation technique dans l'entreprise *blue2i* : une TPE du secteur de l'informatique. Nous verrons comment analyser le contexte et les enjeux de l'entreprise, comment identifier les objectifs et problèmes ciblés et quelle méthodologie mettre en place pour implémenter des solutions adaptées aux besoins et aux contraintes de l'entreprise. Nous verrons également comment mettre en place des éléments d'analyse et d'évaluation pour mesurer l'efficacité des solutions mises en œuvre et pour par la suite faire évoluer ces dernières en fonction des objectifs.

Contexte de l'entreprise, conception et mise en application

Après avoir analysé l'environnement de la tech, effectué des recherches approfondies et émis des hypothèses sur les solutions pouvant répondre aux problèmes de gestion des connaissances et des savoirs-faire, il est maintenant temps de se concentrer sur le cas précis de *blue2i*. La partie suivante permettra d'analyser le contexte spécifique de l'entreprise, d'identifier les besoins et les objectifs ciblés en matière de gestion de documentation technique, et de justifier les choix de solutions retenus. Cette partie fera également le détail de la démarche de mise en œuvre de cette solution. En fin de partie, nous analyserons le retour d'expérience pour évaluer l'efficacité de la solution et identifier les axes d'amélioration possibles.

Contexte, enjeux et objectifs ciblés

Cette partie est consacrée à l'analyse contextualisée de l'entreprise. Elle vise à identifier la problématique spécifique de l'entreprise en matière de gestion des connaissances et des savoirs-faire, ainsi que les enjeux, objectifs et besoins associés. Dans cette optique, il est important de réaliser une analyse de l'environnement existant, afin de comprendre les pratiques et les outils actuellement utilisés dans l'entreprise. Cette analyse permettra de proposer des solutions adaptées aux besoins et contraintes de l'entreprise, et de mettre en place une solution de gestion des connaissances efficace et durable.

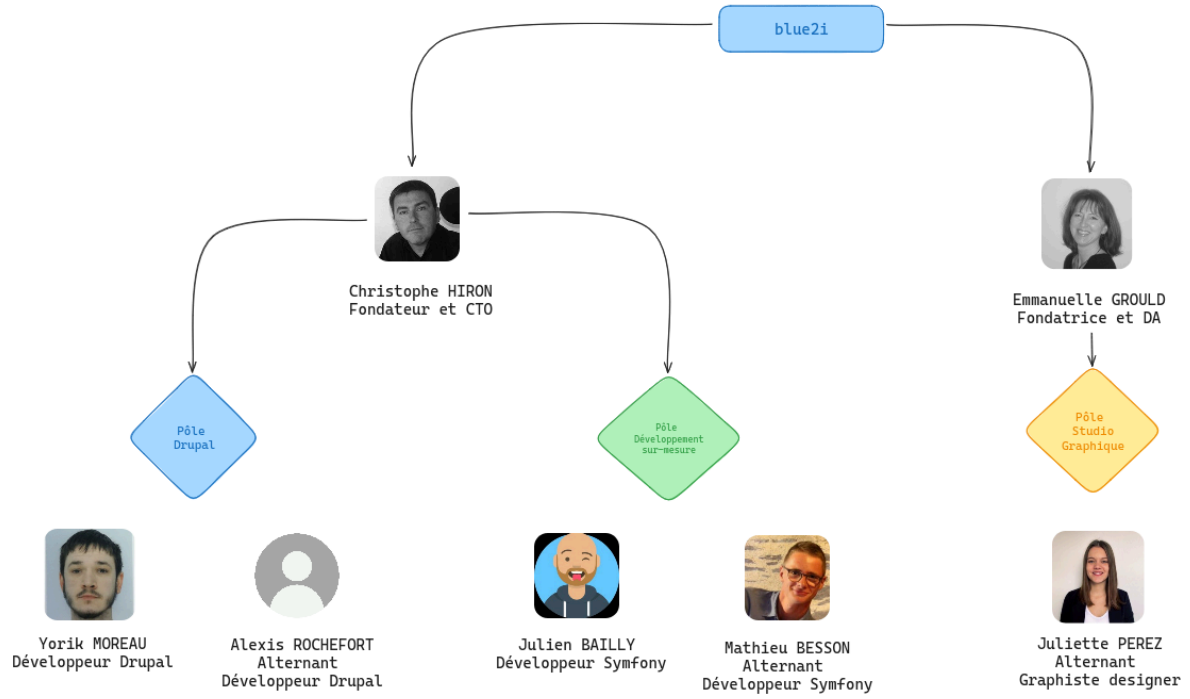
Analyse contextualisé de l'entreprise

Présentation générale de l'entreprise

L'entreprise *blue2i* est une agence web et print (TPE) installée à *Châteaubourg* près de Rennes au 36 Rue Blaise Pascal et a été fondée par *Emmanuelle GROULD* et *Christophe HIRON* en 2010.

Aujourd'hui, l'agence compte 7 collaborateurs regroupés autour de 3 pôles :

- **Studio graphique** : Création de visuels, de maquettes et de supports de communication sur tous formats
- ***Drupal (CMS)*** : Développement de sites "vitrine" ayant besoin d'une administration riche et variée avec le *CMS* (Content Management System) *Drupal*
- **Développement sur-mesure** : Création d'applications sur-mesure avec le framework *Symfony*



Organigramme de blue2i

L'équipe est restreinte et le secteur d'activité est assez varié. Cela amène parfois à l'isolement de certains pôles et potentiellement à une perte de connaissance plus ou moins marquée en cas de départ de collaborateurs. Pour ma part, je suis affecté depuis le début de mon alternance au pôle développement sur-mesure.

Problématique de l'entreprise

La problématique de *blue2i* est liée à la gestion des connaissances. En effet, suite au départ de collaborateurs cadres, l'entreprise fait face à une perte importante de connaissances et de savoir-faire essentiels, complexifiant en conséquence l'intégration de nouveaux collaborateurs et la gestion de nouveaux projets. Les processus et connaissances sont peu ou pas documentés, et l'onboarding de nouveaux collaborateurs est long, fastidieux et coûteux en ressources humaines.

Enjeux

Installé dans un marché ou le turnover est important et où le savoir-faire est une ressource majeure, *blue2i* est aujourd'hui confrontée à de multiples enjeux. Parmi eux, on peut citer la simplification de l'intégration de nouveaux salariés, la limitation de la perte de valeur au départ d'un collaborateur, mais aussi la garantie de la continuité des savoirs critiques de l'entreprise. Pour répondre efficacement à ces enjeux stratégiques importants, il est donc essentiel de mettre en place des solutions garantissant la transmission des savoirs.

Objectifs et besoins

Après avoir explicité les enjeux globaux de la mise en place de cette solution pour *blue2i*, il est nécessaire d'identifier les objectifs et besoins spécifiques de l'entreprise. Les objectifs et besoins sont les suivants :

- Centraliser et organiser l'ensemble de la documentation de l'entreprise pour faciliter leur accès et leur utilisation.

- Améliorer la communication et la coopération entre les différents collaborateurs grâce à une documentation claire et accessible.

- Réduire les risques et les erreurs tout en augmentant la productivité et en fournissant des procédures précises et à jour aux techniciens.

- Faciliter l'intégration des nouveaux salariés en fournissant un guide complet d'intégration sur les processus, les normes et conventions de l'entreprise.

- Préserver les connaissances et les compétences critiques de l'entreprise en documentant en continu les savoir-faire et les bonnes pratiques.

- Permettre la durabilité et la mise à jour de la documentation en utilisant un format efficace et non lié à une solution de visualisation spécifique.

- Structurer la production de documentations techniques en fonction des objectifs et de la cible, tout en assurant sa cohérence et sa qualité.

- Mettre en place une approche de conception de projet basée sur la documentation pour assurer la compréhension commune des attentes et des contraintes du projet par tous les acteurs avant le lancement du développement.

- Gérer la mise à jour de la documentation à l'aide du même processus que la mise à jour du code en gardant le contrôle sur les modifications et en automatisant le déploiement.

De manière à répondre à ces objectifs sur le long terme, il est nécessaire de mener une réflexion approfondie sur l'architecture, les outils et les processus à mettre en place. Les tests et retours d'expérience utilisateurs réguliers sont indispensables pour s'assurer que la solution répond bien aux besoins initiaux et pour apporter les ajustements nécessaires.

Analyse de l'existant, identification des faiblesses et défis

L'environnement organisationnel et technique actuel

Dans le cadre de cette étude, il est important de faire un état des lieux de l'environnement technique et des outils d'organisation utilisés au sein de l'entreprise, en particulier en ce qui concerne la communication et l'organisation. Cette analyse permettra d'identifier les éventuelles faiblesses et axes d'amélioration possibles pour optimiser la transmission des informations et des connaissances.

En septembre 2024, l'organisation de l'entreprise est centrée autour de plusieurs outils :

- ***Nextcloud***[65] pour la rédaction des cahiers des charges, des spécifications techniques et fonctionnelles, des documentations techniques et des documents officiels.
- ***Youtrack***[66] pour la gestion de projets, la gestion des tâches, le support technique, et son API pour la facturation client.
- ***Gitea*** pour les dépôts de version de code source.
- ***Eurecia***[67] pour les ressources humaines.
- ***Team Password Manager***[68] pour la gestion de mots de passe.
- ***Slack***[69] et les mails pour la communication asynchrone et instantanée.

À l'arrivée d'un nouveau projet, le cahier des charges client est remis à plat pour en déduire les spécifications fonctionnelles du projet qui amèneront par la suite à la rédaction des spécifications techniques.

Une fois les spécifications validées par le client et l'agence, elles sont divisées en lots. Ces lots sont ensuite subdivisés pour permettre des livraisons régulières des fonctionnalités. Chaque livraison est suivie par une recette client validant les fonctionnalités réalisées.

Les outils utilisés ne sont pas tous adaptés aux besoins et parfois pauvres en fonctionnalités attendus ce qui, au fil des projets, a mené à une désapprobation générale de ces outils. Les spécifications fonctionnelles/techniques étant sur un outil externe hors du code source du projet, elles sont rarement mises à jour et donc à terme plus en accord avec l'état réel du projet.

---

[65] Nextcloud : Logiciel libre de site d'hébergement de fichiers et une plateforme de collaboration, nextcloud.com/fr/
[66] YouTrack : Logiciel de gestion de projet, suivi de tâches et de gestion de support technique développé par la société JetBrains, www.jetbrains.com/fr-fr/youtrack/
[67] Eurecia : Le logiciel de gestion de ressources humaines, www.eurecia.com/
[68] Team Password Manager : Outil de gestion de mot de passe collaboratif, teampasswordmanager.com/
[69] Slack : application de messagerie instantanée pour les entreprises, slack.com

État des lieux de la documentation technique

Sur la partie technique, chaque développeur tient à jour de son côté ses procédures techniques et notes, qui ne sont par conséquent, pas mutualisées. Les développeurs travaillant pour la plupart individuellement sur les projets, le besoin de mutualisation ne se fait pas ressentir. Les documentations et procédures orientées vers l'administration système sont quant à elles rédigées dans *Nextcloud*. Dans ce cas aussi, loin du code source des projets. Les librairies internes sont elles également très peu, voire pas documentées, les développeurs les utilisant au quotidien, ce qui ne facilite pas l'intégration des nouveaux arrivants.

En 2023, après plusieurs départs dont celui d'un lead développeur, le renouvellement des effectifs soulève des nouvelles problématiques. L'absence de mutualisation et de centralisation de la documentation technique a entraîné plusieurs conséquences négatives pour l'entreprise, parmi elles :

- L'augmentation de la durée d'onboarding (et des ressources humaines associées), de compréhension des normes, d'application des process de l'entreprise et de formation initiale des nouveaux collaborateurs. Cette situation est en partie due à la perte de connaissance et au processus de formation à mettre en œuvre à chaque nouvelle arrivée.
- L'impact significatif sur les performances des collaborateurs amnène à un ralentissement global des performances de l'entreprise.
- Les procédures et méthodologies ne sont pas documentées amenant à une augmentation des anomalies techniques.
- La transmission des besoins client aux collaborateurs et la gestion des projets sur le long terme est difficile.

Ces conséquences peuvent être attribuées à plusieurs facteurs, tels que :

- Les procédures et connaissances sont peu, voire pas documentées, ou loin des sources des projets (voire individualisés), créant une charge mentale forte pour les mises à jour.
- Une perte globale de savoir-faire suite aux départs de collaborateurs cadres de l'entreprise.
- La difficulté de transmission des besoins et des spécifications et un suivi de projets complexe et inopérant.

Analyse et bilan de l'environnement actuel

L'état des lieux de l'environnement technique et de l'organisation de l'entreprise révèle plusieurs faiblesses dans la gestion des connaissances et des projets. Tout d'abord, l'utilisation d'outils de documentation inadaptée comme *Nextcloud* entraîne une désapprobation générale, un frein à la rédaction et à la mise à jour des documentations et une perte de productivité générale. Associée à l'isolation des différents pôles, cette

contrainte pousse les collaborateurs à moins documenter leurs processus entraînant une perte importante de connaissances communes.

Ensuite, la documentation technique n'est pas commune ou souvent très éloignée du code source du projet, ce qui entraîne une perte de connaissances techniques et de procédures non documentées ou non mutualisées au sein de l'entreprise. Les départs de plusieurs collaborateurs, dont celui d'un lead développeur, ont amplifié ces problématiques en augmentant la durée d'onboarding, de compréhension et d'adoption des normes en prolongeant significativement la durée de formation initiale des collaborateurs.

Pour finir, des difficultés de gestion, de suivi de temps et de transmission des besoins client aux collaborateurs ainsi qu'une augmentation des anomalies techniques ont également été constatés. Ces éléments ont eu un impact significatif sur les performances des collaborateurs et ont eu pour conséquence un ralentissement global de la productivité de l'entreprise.

Risques du déploiement de nouvelles solutions

Durant la phase d'établissement de ces nouvelles solutions, plusieurs risques et événements redoutés peuvent potentiellement survenir. Ceux-ci pourraient influencer sur sa mise en application et sur ces bénéfices :

- **Changements inattendus des besoins** : Lors de l'implémentation des recommandations, des modifications imprévues ou non prises en compte des besoins de l'entreprise peuvent perturber la mise en œuvre initiale en engendrant des évolutions importantes et un allongement de la durée de mise en place.
- **Contraintes de ressources humaines et budgétaires** : En termes de budget, de temps ou de main-d'œuvre, des restrictions de l'entreprise pourraient impacter la préparation des différents éléments du projet en entraînant des compromis ou des ajustements nécessaires. En effet, des échéances importantes en interne seraient un facteur de réduction du temps alloué à l'établissement de nouveaux outils.
- **Limitations technologiques** : Des problèmes liés aux technologies utilisées, comme des limitations inattendues, des contraintes techniques ou encore des difficultés de mise en place sont à redouter. La phase de validation technique et d'établissement du *POC* (Proof of Concept) est donc importante pour identifier les outils adéquats en fonction des besoins.
- **Résistance au changement** : Les membres de l'équipe pourraient être réticents à une évolution de leurs outils qui viendrait bouleverser leurs habitudes. En effet, cet élément pourrait compromettre l'adoption de nouvelles solutions.
- **Écart entre résultats de la recherche et besoins de l'entreprise** : En cas de désaccords significatifs entre les résultats de recherche et les besoins spécifiques de l'entreprise, des adaptations majeures pourraient compromettre la pertinence du projet.

Conception et planification de la solution : Choix techniques, architecture et budget

Dans la partie précédente, nous avons identifié les besoins spécifiques de *blue2i* en matière de gestion de la documentation. La partie suivante se concentrera sur la justification des choix techniques, l'explication de l'architecture mise en place et le budget estimé de la création jusqu'au déploiement de la solution.

Justification des choix techniques : Des recherches préalables vers la mise en oeuvre

Cette partie vise à justifier les choix techniques établis en fonction des besoins préalablement identifiés. Il peut donc être intéressant de partir des besoins établis pour en identifier les éléments techniques ou organisationnels les plus adaptés. L'objectif est de fournir une justification claire et détaillée des choix techniques, tout en énonçant les critères de sélection des outils sélectionnés.

Markdown : Un langage de markup efficace et durable

Dans le cadre de la rédaction de documentation technique, il est essentiel de faire le bon choix lors de la sélection de son outil de rédaction. Dans le contexte de *blue2i*, les docs sont rédigés par les développeurs eux-mêmes ou par le responsable technique.

Après analyse des différentes options, le choix s'est porté sur le format *Markdown* comme langage de markup léger de rédaction de documentation. Ce choix répond particulièrement bien aux objectifs établis :

- **Durabilité et dissociation des outils** : Le *Markdown* est un format de texte brut qui ne dépend pas d'un logiciel ou d'une plateforme spécifique. Il est donc facilement transférable et lisible sur n'importe quel système, ce qui garantit la durabilité de la documentation dans le temps y compris en cas de changement d'outil de visualisation.

- **Simplicité et intuitivité** : Les fonctionnalités du langage étant limitées et restreintes, le rédacteur se concentre plus sur le contenu que sur la forme. Cette simplicité facilite aussi la lecture du texte brut sans outil de mise en forme.

- **Efficacité** de **mise à jour** : Il s'agit d'un langage de balisage léger qui permet une mise en forme rapide et efficace du contenu. À l'intervention sur un projet, le développeur peut donc rapidement mettre à jour la documentation au plus près du code source du projet sans perte d'efficacité. Ce format étant facilement lisible depuis n'importe quel éditeur de code, chaque documentation est directement encapsulée au plus proche du code source lié.

- **Compatibilité** : Étant un langage populaire et répandu, l'utilisation de *Markdown* garantit une grande compatibilité avec différents outils. À partir de ce format il est donc facilement possible d'exporter un contenu stylisé et mis en forme automatiquement avec de nombreux outils vers des formats comme HTML, PDF, EPUB. De plus, sa syntaxe simple et identifiable le rend aisément parsable par des outils d'analyse.

En conclusion, le choix d'utiliser le format *Markdown* pour notre solution de gestion de la documentation répond aux objectifs établis en termes de durabilité, efficacité, simplicité et compatibilité.

MkDocs Material : Outil de build d'un wiki ergonomique à partir du format Markdown

Dans le cadre de l'utilisation d'un outil de markup léger, il est important de déterminer quel outil de mise en forme utiliser pour faciliter l'accès et la visualisation de la documentation produite. Les besoins clés étaient la centralisation et la facilitation de l'accès à la documentation, une ergonomie UX (User Experience) et DX (Développer experience) efficace, une solution open-source et auto-hébergée.

Après étude des différentes options, le choix s'est porté sur *MkDocs Material* comme outil de build de markdown pour répondre aux besoins de l'entreprise. Ce choix répond aux besoins établis :

- **Centralisation et facilité d'accès** : *MkDocs* permet de créer un wiki en ligne de référence pour toute la documentation de l'entreprise, accessible depuis un seul et même endroit.

- **Ergonomie UX et DX** : Cet outil offre une UI/UX moderne et efficace pour les utilisateurs finaux, ainsi qu'une expérience idéale pour les développeurs car proche des documentations d'outils classiques du milieu. La prise en charge de la coloration syntaxique de nombreux langages et les multiples fonctionnalités additionnelles de mise en forme (sommaire intéractif, recherche avancée, tags…) et de personnalisation font de cet outil l'outil idéal.

- **Solution open-source et auto-hébergée** : Étant open-source, *MkDocs* est totalement gratuit et en évolution continue grâce à sa communauté de contributeurs active. De plus, l'installation auto-hébergée permet un contrôle complet sur les données et la personnalisation. Il est ainsi facilement possible de développer des modules complémentaires en étendant les fonctionnalités initiales de l'outil.

Le choix d'un outil de build de *Markdown* vers un wiki statique s'est donc porté vers *MkDocs Material* car il répond aux besoins clés établis en matière de centralisation, d'ergonomie et de fonctionnalités. De plus, par son aspect open-source, cet outil est totalement ouvert à la personnalisation et à l'extension de ses fonctionnalités de base. Pour

les fonctionnalités manquantes sur l'outil, comme l'agrégation de sources, la détection des liens cassés externes ou encore l'automatisation des build, nous avons choisi d'utiliser la stack technique habituelle de l'entreprise avec le framework *Symfony*.

Git : Une approche de gestion de production logiciel pour la documentation

Pour assurer une documentation de qualité, fiable et à jour, il est important de mettre en place un processus de gestion efficace. Dans cette optique, nous avons choisi d'utiliser un processus de gestion de la documentation similaire à celui utilisé pour la gestion du code source avec des outils tels que *git* et *Gitea*. Les raisons de ce choix sont associées aux besoins suivants :

- **Processus de production et de contrôle** : L'utilisation d'un gestionnaire de version tel que *git* pour la documentation, permet, au-delà de l'historisation, de garder un contrôle sur la documentation produite à travers des fonctionnalités tel que les pull-request. De plus, cette méthode de gestion permet aussi d'automatiser facilement le déploiement grâce à des webhooks *Gitea* par exemple.
- **Documentation au plus près du code** : En stockant la documentation dans les dépôts git de projet ou dans des dépôts dédiés, la documentation peut être maintenue au plus près du code, limitant les frictions à la mise à jour et encourageant les équipes à maintenir la documentation à jour.

En conclusion, l'utilisation d'un processus de production et de contrôle similaire à celui du code source pour la gestion de la documentation offre des avantages importants, qui, auront un impact indéniable sur la productivité des équipes.

Diataxis et Documentation-Driven Design : Des concepts pour cadrer la rédaction documentaire

De manière à organiser au mieux la production de documentation, il peut être intéressant de mettre en place des cadres de travail efficaces pour garantir la qualité, la pertinence de la forme et l'accessibilité de l'information. Ainsi, l'utilisation d'outils comme *Diataxis* ou le *Documentation-Driven Design* peuvent apporter ce cadre permettant d'atteindre les objectifs de l'entreprise :

- **Structurer la rédaction** en établissant la **cible** et le **moyen** : Diataxis fournit un cadre de travail qui permet de structurer la rédaction en fonction de la cible et du moyen et en divisant la documentation en quatre types : tutoriels, guides pratiques, explications et références. Pour chaque type, il offre des conseils concrets sur la façon de rédiger, de structurer et de présenter l'information pour répondre aux

besoins spécifiques de l'utilisateur. L'utilisation de ce framework permet aussi d'assurer la cohérence de style dans les documentations.

- Éviter les **erreurs d'incompréhension** au lancement d'un projet : Documentation-Driven Design permet de clarifier les exigences et les attentes du projet dès le début, en rédigeant des spécifications techniques et fonctionnelles détaillées, ce qui réduit les risques de malentendus et d'erreurs.

L'utilisation des concepts de *Diataxis* et de *Documentation-Driven Design* permettent de structurer la rédaction en apportant de la qualité, de la pertinence et de l'accessibilité à la documentation, tout en répondant efficacement aux besoins utilisateurs.

Analyse de l'architecture

L'objectif de cette partie est de décrire l'architecture de la solution mise en place pour la gestion de la documentation afin de répondre aux objectifs de *blue2i*. Le schéma ci-dessous décrit les étapes et la structure des éléments en interaction lors de la construction d'un wiki statique à partir de sources de documentations multiples.

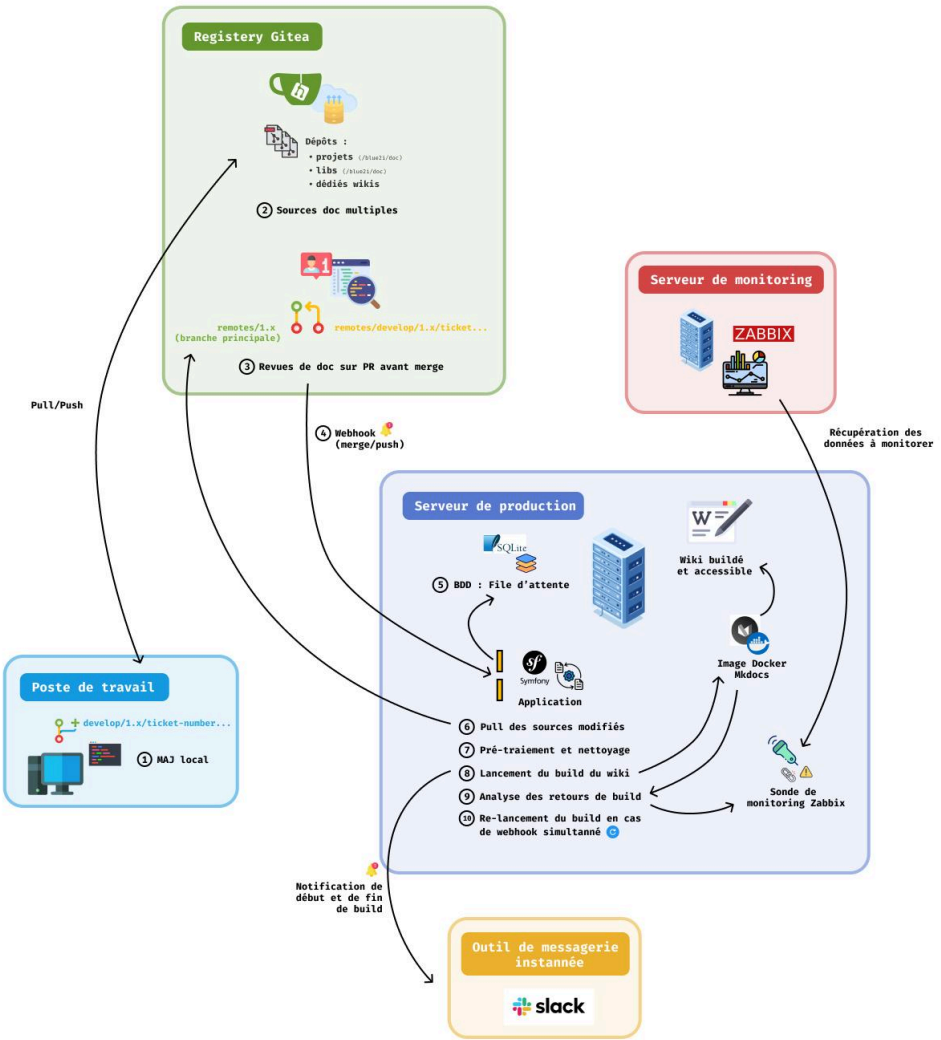


Schéma d'architecture de la solution mise en oeuvre (taille maximale disponible en annexe 4)

Les étapes détaillées sur ce schéma permettent de mieux comprendre la gestion et le processus de construction d'un wiki statique à partir de documentations en *Markdown*. Les points ci-dessous permettent de détailler les différentes étapes du processus : allant des revues de doc par pull request jusqu'à la construction finale du wiki :

1. **Mise à jour locale des sources de documentation** : les développeurs travaillent sur les projets sur leurs machines et apportent des modifications aux sources de documentation sur la branche liée à leur ticket tel que : *develop/1.x/ticket-number-dénomination-branche*
2. **Sources de doc multiples** : les sources de documentation sont stockées dans différents dépôts git, tels que les dépôts de code de projet, de librairies et de documentations dédiés. Les dépôts dédiés à la documentation sont organisés en fonction de leur forme (cadre *Diataxis* adapté), tels que les procédures, les connaissances ou encore les formations. Dans les dépôts de code, les sources de documentation sont ensuite normées dans un dossier /blue2i/doc à la racine du dépôt.
3. **Contrôle par revue de doc** : Après avoir apporté des modifications à leur branche locale, les développeurs push leur branche vers le dépôt *Gitea* et créent une demande de pull-request. Le *CTO* ou responsable du dépôt examine la demande d'intégration et fournit des demandes de modification si nécessaire. Une fois les modifications validées, la branche de travail est mergée sur la branche principale.
4. **Webhooks (notifications) de push sur les dépôts** : Une fois la *PR* (pull request) accepté et mergée dans la branche principale, *Gitea* émet un webhook (configuré sur les push et les fusions sur la branche principale) sur une route de l'application *Symfony* avec les informations du dépôt modifié.
5. **File d'attente** : L'application *Symfony* gère une file d'attente des dépôts à mettre à jour dans une base de données *SQLite*[70]. Un seul processus de build peut être lancé en simultané. Une notification *Slack* de début de processus de build est envoyée.
6. **Récupération des sources modifiées** : Les dépôts modifiés sont mis à jour sur le serveur de production pour obtenir la dernière version de la branche principale.
7. **Pré-traitement et nettoyage des sources de documentation** : Le code source est nettoyé et organisé pour ne contenir que la documentation avec une architecture normalisée pour le futur build avec *MkDocs*.
8. **Build du wiki final depuis les sources** : Le wiki final est construit à partir des sources de documentation sur le serveur.
9. **Analyse des retours de build** : Le retour de build de wiki est ensuite analysé pour identifier les anomalies de build tel que les liens internes cassés ou encore les anomalies bloquantes ou limitantes. Ces anomalies sont ensuite envoyées dans la sonde de monitoring pour notifier les responsables à travers l'outil de monitoring de l'entreprise : *Zabbix*[71].
10. **Notification de mise à jour terminée** : La référence de modification du dépôt est ensuite supprimée de la base *SQLite* et une notification *Slack* est envoyée.

[70] SQLite : Moteur de base de donnée basé sur le langage SQL et basé sur un système de gestion sous forme de fichier, sqlite.org/
[71] Zabbix : Logiciel open-source de monitoring de serveur et d'outils réseaux, www.zabbix.com/

11. **Re-lancement du processus en cas de webhook simultanés** : Une fois le build terminé, on vérifie la base de données *SQLite* et on rejoue le processus si des notifications de mise à jour ont été envoyées durant le build.

En résumé, l'architecture de cette solution de gestion de la documentation utilise des dépôts git pour stocker les sources de documentation, des webhooks pour notifier notre application *Symfony* des mises à jour, et un script *PHP* pour compiler et organiser les ressources afin de créer le wiki final. Des processus d'analyse de build et de monitoring ont également été mis en place pour assurer la fiabilité et la qualité de la solution.

Estimation budgétaire

Dans le cadre du développement d'une nouvelle solution, il est important d'estimer et de chiffrer la charge de travail à fournir pour arriver au résultat attendu. En effet, ce point permet de mieux planifier les différentes étapes du projet, de définir les ressources nécessaires et d'anticiper les coûts estimatifs du projet. Cette première estimation sert de référence lors du développement du projet comme suivi de dépense et d'avancement. Le tableau suivant résume l'estimation budgétaire associée au développement de cette solution :

Tableau de budget estimatif complet en annexe 5.

Récapitulatif du tableau de budget estimatif :

Phase	Tâche	Durée
Conception	Validation des choix techniques et définition de l'architecture logicielle Définition des tâches, objectifs et jalons Développement et tests initiaux du POC	59h
Production	Mise à jour et agrégation des sources de documentation Configuration des webhooks et pré-traitement des sources Compilation, réorganisation des ressources et build du wiki final	69h
Livraison	Déploiement et automatisation de la solution Analyse des retours utilisateurs et correction des bugs Mesure de l'efficacité et planification des évolutions futures Formation de maintenance en interne	30h

Maintenance (mensuel)	Correction de bugs Mise à jour des outils	5h
Total estimé		163h (23j)

L'estimation du budget pour le lancement de la solution de gestion de documentation comprend trois grandes étapes : la conception, la production et la livraison. Cette estimation de budget permet de prévoir les coûts et les ressources nécessaires à la réalisation du projet mais aussi de planifier les différentes phases de la réalisation.

Méthodologie et mise en oeuvre

Après avoir justifié les choix techniques et d'architecture dans la partie précédente, cette dernière partie analyse la méthodologie mise en œuvre pour mettre en place la solution identifiée. Ainsi cette partie présentera la gestion de projet mise en application, les grandes phases et étapes du projet, l'analyse des premiers résultats, réussites et échecs et pour finir les perspectives d'évolution et axes de poursuite du projet.

Présentation de la gestion de projet

Dans le cadre de ce projet, *blue2i* a fait le choix de s'orienter sur la méthode de gestion de projet en cycle en V. Cette méthode est adaptée au contexte de l'entreprise sur un projet interne restreint, où les besoins et exigences définis préalablement sont clairs et bien établis et où l'équipe de développement n'est composée que d'un seul et unique développeur.

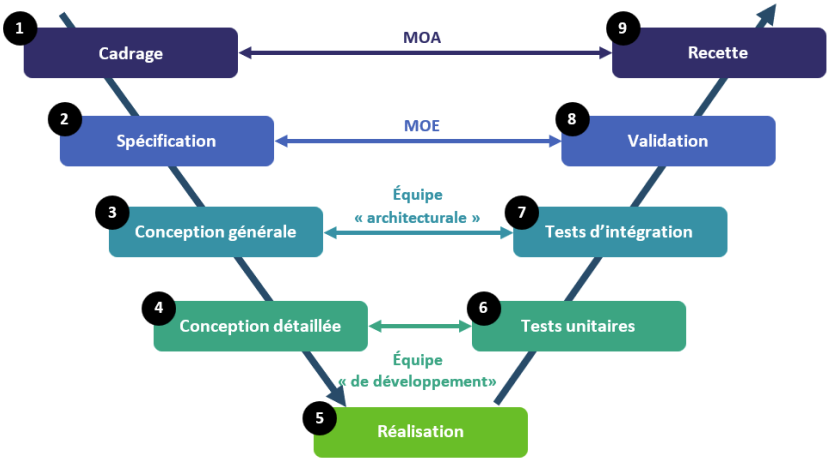


Schéma explicatif de la méthode de gestion de projet en cycle en V[72]

[72] Laetitia Narçon, Cycle en V vs Agile : le match !, 14 Juin 2023, findle.fr/cycle-v-vs-cycle-agile/

Les acteurs principaux de ce projet sont l'équipe de développement, en l'occurrence un seul développeur (moi-même), le "validateur" CTO de *blue2i* Christophe HIRON et les utilisateurs finaux représentés par les développeurs de l'agence.

Malgré l'aspect linéaire et l'effet tunnel lié à cette méthode de gestion de projet, des points d'étapes réguliers sont prévus à chaque fin d'étape clé du cycle et à chaque fin de développement d'une macro-fonctionnalité. Ces points de vérifications permettent de constater si le plan de route est bien valide et si les objectifs intermédiaires sont atteints. Ils permettent aussi de détecter rapidement tout écart ou problème et de prendre des mesures correctives si nécessaire.

Le méthode de gestion de projet en cycle en V se caractérise par une série de phases consécutives avec pour chacune un rôle bien précis. Les étapes sont les suivantes :

1. **Étude des besoins initiaux** : Comprendre les besoins et les attentes des utilisateurs finaux, afin de définir les fonctionnalités à développer.
2. **Spécifications fonctionnelles** : Décrire précisément les fonctionnalités à développer, en utilisant un langage compréhensible par tous les membres de l'équipe.
3. **Conception technique et architecturale** : Définir l'architecture technique de la solution, en prenant en compte les contraintes techniques et les choix technologiques.
4. **Conception détaillée** : Détailler les différents composants de la solution, en précisant leur rôle et leur fonctionnement.
5. **Réalisation / Développement** : Développer et implémenter les différents composants de la solution.
6. **Tests unitaires** : Tester chaque composant de manière isolée, afin de vérifier son bon fonctionnement. Dans notre cas, faute de temps et de moyens, nous n'avons pas mis en place de tests unitaires.
7. **Test d'intégration** : Tester l'ensemble des composants de la solution, afin de vérifier leur bonne interaction. Dans notre cas, faute de temps et de moyens, nous n'avons pas mis en place de tests d'intégration.
8. **Tests fonctionnels** : Tester la solution du point de vue de l'utilisateur final, afin de vérifier qu'elle répond bien à ses besoins et attentes.
9. **Recette / Validation finale** : Valider définitivement la solution, en s'assurant qu'elle est conforme aux spécifications fonctionnelles et techniques.
10. **Déploiement** : Mettre en production finale la solution.

Par manque de moyens et de temps, *blue2i* a choisi de ne pas mettre en place de tests unitaires et d'intégration dans la mise en œuvre de la solution. Malgré tout, des tests fonctionnels manuels ont été réalisés de manière à assurer que la solution répond aux besoins et aux exigences définis initialement. Ces tests fonctionnels sont essentiels pour valider la solution et s'assurer qu'elle est prête à être déployée. Pour ma part, j'aurais préféré pouvoir mettre en place des tests unitaires et d'intégration dans le but d'assurer la qualité et la sérénité des déploiement de la solution.

Étapes de mise en oeuvre de la solution

Dans le cadre du déploiement de cette solution, il est intéressant de décrire les étapes successives ayant guidé le projet. Dans cette partie, nous allons identifier en détails chacune de ces étapes, en partant de l'étude des besoins initiaux jusqu'à la livraison finale du projet en production.

Étude des besoins initiaux

Durant cette première étape d'étude et d'analyse des besoins, plusieurs actions ont été mises en place pour comprendre au mieux les attentes de l'entreprise en matière de gestion de la documentation.

Tout d'abord, nous avons établi les objectifs de la solution lors d'une réunion d'"identification des besoins" de la solution finale. Cette réunion a eu lieu entre le CTO de l'entreprise et moi-même. Elle a notamment permis de définir les objectifs et les enjeux de l'implémentation d'une solution de gestion de la documentation, ainsi que les fonctionnalités attendues.

Ensuite, une étude des différentes solutions et outils de gestion de la documentation existants sur le marché a été réalisée. Cette étude a permis de comparer les différentes solutions et de sélectionner celles qui répondaient le mieux aux besoins de l'entreprise. Un des objectifs était aussi de répertorier les bonnes pratiques du secteur sur le sujet. Une enquête a également été menée auprès de l'équipe pour recueillir leurs avis et leurs pratiques individuelles en matière de gestion de la documentation, autant en entreprise qu'en dehors. Cette enquête a permis de comprendre les attentes et les besoins de chacun et d'identifier les éventuels points de friction dans la gestion et la rédaction de la documentation.

Au terme de cette phase d'étude, plusieurs solutions open-source de build pouvant répondre au besoin ont été répertoriées tel que *Wiki.js*, *Docusaurus* ou encore *MkDocs Material*. Finalement, après comparaison des fonctionnalités, le choix s'est porté sur *MkDocs* à la fin de cette phase.

Spécifications fonctionnelles, techniques et conception détaillée

Les étapes suivantes sont la rédaction des spécifications fonctionnelles et techniques, ainsi que la conception détaillée.

Les spécifications fonctionnelles ont été élaborées en prenant en compte les besoins initiaux étudiés lors de la première étape. Ces spécifications détaillent les fonctionnalités attendues de la solution, ainsi que les contraintes et les exigences auxquelles elle doit répondre.

En parallèle, la conception technique et structurelle de la solution permet d'identifier les caractéristiques techniques du projet comme les outils ou logiciels choisis. C'est dans ce

document que les rôles et interactions des différents outils utilisés dans la solution sont détaillés. Après la rédaction de ces deux documents, nous avons défini l'architecture globale du projet de manière à identifier chaque étape et interaction entre différents composants.

Les spécifications fonctionnelles et techniques sont disponibles en annexes 6 et 7.

À ce stade, nous avons réalisé un premier *POC* de la solution finale pour vérifier et valider les principaux attendus. Ce *POC* à notamment permis de vérifier les éléments suivant sur la solution sélectionnée :

- la possibilité d'agrégation de sources multiples en markdown
- un build rapide et une détection des erreurs efficace
- une installation auto-hébergée et une gestion des mises à jour efficaces
- la possibilité de personnalisation visuelle importante et une prise en charge de la coloration syntaxique des langages
- une fonctionnalité de recherche avancée
- une architecture de wiki efficace et intuitive
- une UI/UX moderne et efficace
- un sommaire de document

Enfin, une fois le *POC* validé, le projet à été découpé en plusieurs tâches à réaliser sur *Youtrack* (l'outil de gestion de projet de l'entreprise) et des objectifs d'étapes à atteindre ont aussi été définis. Ces éléments permettent de planifier la réalisation de la solution et de suivre son avancement.

Développement et tests fonctionnels

L'étape suivante est la réalisation technique du projet. Durant cette phase de développement, l'équipe technique implémente les fonctionnalités identifiées durant les étapes précédentes.

Cette étape nécessite l'installation, la configuration d'outils et les développements suivants :

- Installation de l'image docker de *MkDocs Material* : *squidfunk/mkdocs-material*
- Configuration des webhooks (notifications) de push sur les dépôts sources
- Agrégation des sources multiples
- Mise à jour locale des sources de documentation
- Pré-traitement et nettoyage des sources de documentation
- Compilation et ré-organisation des ressources
- Build du wiki final depuis les sources avec l'image docker en construisant un conteneur de build temporaire
- Analyse des retours de build : identification des anomalies bloquantes ou limitantes
- Notification en temps réel de la mise à jour de la production (et des sources mises à jour)
- Création des sondes de monitoring
- Développement d'un outil de vérification de lien externes cassés

Cette phase est suivie par la phase de test, ici *blue2i* a choisi de mener seulement des tests fonctionnels manuels pour valider la solution avant un déploiement en production.

```
[global]
publishFolder = 'racine/fr'
mdFolder = 'racine/md'

# Liste des dossiers à aplatir (récursivement dans toute l'arbo)
flattenFolderList[]= "blue2i/doc"
flattenFolderList[]= "src"

# Liste des dossiers à supprimer (récursivement dans toute l'arbo)
removeFolderList[]= "blue2i/checklist"
removeFolderList[]= "blue2i/suivi-projet"
removeFolderList[]= "b2ifmkconf.d"
removeFolderList[]= "b2ilibext"
removeFolderList[]= "libext"
removeFolderList[]= "html_include"

[src]
# Dépôts organisation wiki
repository[] = "ssh://b2iinstall@gitea.blue2i.com:2222/wiki/connaissancestechniques.git;connaissances/techniques"
repository[] = "ssh://b2iinstall@gitea.blue2i.com:2222/wiki/conventionstechniques.git;conventions/techniques"
repository[] = "ssh://b2iinstall@gitea.blue2i.com:2222/wiki/formations.git;formations"
repository[] = "ssh://b2iinstall@gitea.blue2i.com:2222/wiki/procedurestechniques.git;procedures/techniques"
repository[] = "ssh://b2iinstall@gitea.blue2i.com:2222/wiki/proceduresdiverses.git;procedures/divers"

# ...
```

Extrait des configurations du wiki permettant de configurer l'application (dépôts de code source, dossier source/cible de build…).

Livraison, maintenance et évolutions

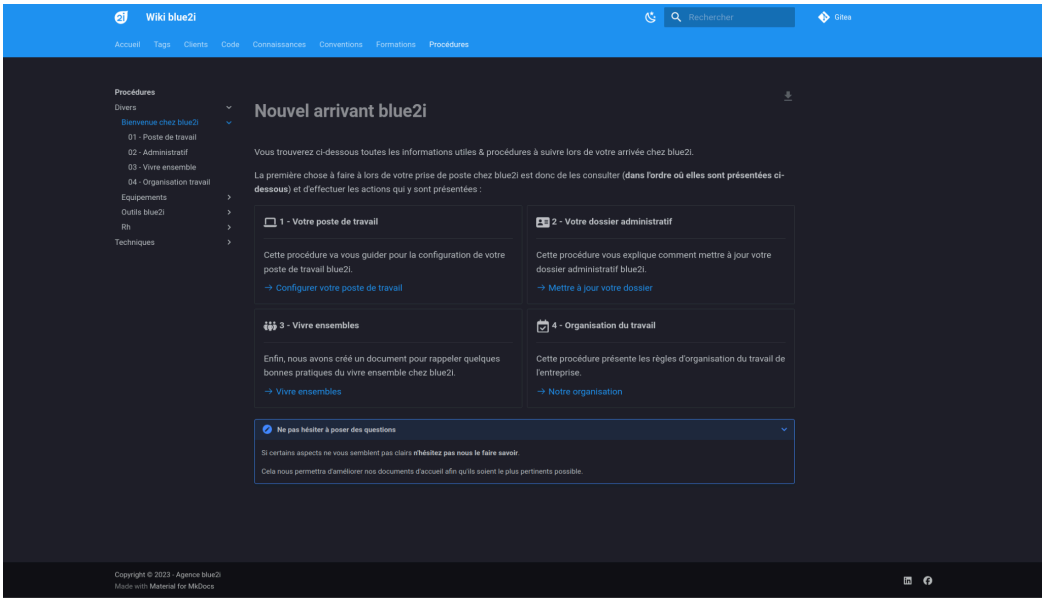
Les dernières étapes pour le déploiement de la solution sont la livraison, la maintenance et les éventuelles évolutions à venir. Ces étapes permettent de clôturer la phase de build du projet afin d'obtenir les premiers retours utilisateurs.

Une fois tous les outils configurés et les fonctionnalités sur-mesure développés en environnement de développement, il est maintenant temps de passer l'application en production. Le déploiement de l'application est géré grâce à un outil en ligne de commande, interne à l'entreprise, permettant de déployer nos applications en production : *b2iinstall*. Cet outil permet de semi-automatiser le déploiement en installant sur un serveur l'application en mode production, ses dépendances et en configurant certains éléments spécifiques de l'environnement de déploiement.

Une fois la solution déployée, les premiers retours utilisateurs des collaborateurs sont analysés avec attention pour apporter des ajustements si nécessaire. Nous avons ensuite réalisé une réunion de clôture permettant de faire le bilan du projet et d'identifier les perspectives d'évolution à moyen et long terme.

J'ai ensuite pu effectuer une présentation de l'outil et de ses fonctionnalités auprès des utilisateurs finaux, en particulier les développeurs, pour garantir la bonne compréhension de la solution mise en place. De plus, cette formation m'a aussi permis de transmettre des informations sur les bonnes pratiques de rédaction de la documentation, telles que *Diataxis* ou *Documentation-Driven Design* pour assurer la qualité et la pérennité des documentations.

La maintenance et la mise à jour de l'image *MkDocs* et du projet sont assurées régulièrement pour garantir la sécurité et la performance de la solution. Une veille sur l'outil et ses nouvelles fonctionnalités nous permet aussi de faire évoluer nos pratiques avec l'outil et d'envisager des axes d'améliorations.



Wiki final après build des fichiers Markdown

Analyse des résultats, réussites et échecs

Cette partie présente l'analyse des résultats obtenus à la suite au lancement de la solution. Dans un premier temps, il est intéressant de s'arrêter sur les différents indicateurs choisis pour mesurer l'efficacité de notre solution. Ensuite, nous nous attarderons sur les réussites constatés grâce à la solution. Enfin, nous analyserons les échecs rencontrés, en identifiant leurs causes et en proposant des solutions envisageables pour y remédier.

Indicateurs

Après la mise en place de la solution de gestion de documentation, un sondage a été réalisé au sein de l'entreprise. Les résultats ont montré que les salariés étaient en moyenne tous satisfaits de la nouvelle solution. Ils ont trouvé la prise en main de l'outil facile et ont apprécié la qualité et l'accessibilité des informations, ce qui leur a permis de gagner du temps dans leur travail quotidien.

Le sondage et les résultats sont disponibles en annexe 8.

Pour la mise à jour des documentations, sur une période de deux mois, les quatre développeurs pouvant modifier les documentations ont effectué en moyenne au total sept

builds par jour. Cela montre que la solution est utilisée de manière régulière et que les mises à jour sont fréquentes.

Le temps moyen de déploiement d'une nouvelle version du wiki est de 3 minutes et 30 secondes à partir de la validation d'une *PR* (pull request). Cela permet une mise à disposition rapide des nouvelles informations pour l'ensemble des collaborateurs.

En termes de maintenance, les liens cassés externes ou internes sont détectés dans la journée grâce à l'outil de détection et sont corrigés dans les deux jours suivants. Cela garantit une qualité de contenu élevée et une navigation fluide pour les utilisateurs.

Enfin, l'intégration de nouveaux salariés est grandement facilitée par la solution de gestion de documentation. En effet, en seulement deux jours, le nouveau salarié peut suivre le processus d'intégration de l'entreprise à travers la lecture du wiki et les démarches administratives. Ensuite, en un à deux mois, il comprend l'ensemble des conventions, normes et processus de l'entreprise de manière quasi autonome, ce qui réduit considérablement les efforts conséquents d'onboarding pour les autres collaborateurs et facilite son intégration.

Réussites

La mise en œuvre de la solution de gestion de documentation à permis de répondre aux objectifs établis initialement. Tout d'abord, chaque mise à jour de documentation est désormais soumise à une revue de code, ce qui permet d'obtenir des retours constructifs pour améliorer la qualité de la documentation. Les nouveaux projets sont maintenant systématiquement précédés de la rédaction de spécifications techniques et fonctionnelles, ce qui permet de bien définir les besoins et les attentes de chaque acteur du projet.

Ensuite, l'utilisation d'un langage de balisage léger pour la rédaction de la documentation a permis de garantir la durabilité et la pérennité des informations, même en cas de changement d'outil. De plus, l'ensemble des procédures techniques sont désormais totalement documentées dès les premières étapes de recherche sur un nouvel outil, ce qui permet de ne perdre aucune valeur en cas de départ d'un salarié.

Enfin, la centralisation de l'ensemble des documentations techniques au sein du wiki permet de faciliter l'accès et la recherche d'informations pour l'ensemble des collaborateurs. Les nouveaux salariés peuvent également être intégrés beaucoup plus rapidement et avec bien moins d'efforts pour les autres collaborateurs, grâce à un processus d'intégration clair et documenté au sein du wiki pouvant être suivi de manière quasi autonome.

Échecs et solutions envisagés

Tout d'abord, certaines fonctionnalités de visualisation comme les blocks d'alerte ou encore les badges du wiki ne sont accessibles que par des syntaxes spécifiques à *MkDocs*. Le choix d'utiliser ces fonctionnalités pourrait rendre un éventuel changement d'outil contraignant. Cependant, l'entreprise est consciente de cette contrainte et a choisi de prendre ce risque afin de bénéficier de toutes les fonctionnalités offertes par *MkDocs*.

Il y a très peu de documentation sur la partie studio graphique. Pour remédier à cela, il pourrait être intéressant de mettre en place un éditeur *WYSIWYG* permettant aux collaborateurs du studio de pouvoir rédiger de la documentation de manière simple et intuitive avec une visualisation en direct du résultat. Cette fonctionnalité permettrait d'inclure véritablement ce pôle dans le processus de documentation.

Enfin, les codes blocks des applications ne sont pas encore intégrés au wiki. Pour résoudre ce problème, des solutions telles que *PhpDocumentor*, qui permettent de construire dynamiquement de la description de code, pourraient être envisagées. Cette solution permettrait d'intégrer la documentation de code au sein même du wiki, dans une section dédiée servant de référence du code source.

En conclusion, l'analyse des indicateurs, des succès et des échecs de la mise en place de la solution permet d'identifier les objectifs atteints et les axes d'amélioration et d'évolution de la solution. En global, la solution répond au besoins et attentes initiales de l'entreprise en offrant un point d'accès de toute la documentation de l'entreprise en un seul et même outil.

Perspectives d'évolution et axes de poursuite

Pour finir, dans la perspective d'amélioration continue de notre solution de gestion de documentation, nous avons identifié plusieurs axes d'évolution.

Tout d'abord, nous souhaitons nous pencher sur la documentation par le code. Les recherches préalables ont mené vers l'utilisation de *PhpDoc (PhpDocumentor)* et d'une surcouche avec rendu markdown *(Saggre/phpDocumentor-markdown[73])* pour créer de la documentation de code source au format markdown et l'intégrer dans notre wiki. Cela permettrait de centraliser encore davantage l'ensemble de nos documentations techniques.

Ensuite, une des perspectives d'évolution envisagée est l'intégration d'une intelligence artificielle *LLM* (Language Learning Model)[74] pour en faire un assistant personnel. L'objectif serait de lui transmettre l'ensemble de notre documentation pour qu'il puisse répondre aux questions des collaborateurs de manière autonome et ainsi faciliter l'accès à l'information.

Nous souhaitons également à plus ou moins long terme mettre en place un outil de recherche global, tel qu'*Elasticsearch*[75], pour permettre de rechercher dans l'ensemble de nos outils : le code source, la documentation du wiki, les fichiers du cloud, *Youtrack*, etc. Cela permettrait de gagner du temps et d'améliorer l'efficacité de nos recherches pour accéder encore plus vite à l'information.

---

[73] Librairie open source basé sur PhpDocumentor permettant la conversion au format markdown : github.com/Saggre/phpDocumentor-markdown

[74] LLM (Language Learning Model) : Modèle d'ia de langage

[75] Elasticsearch : Moteur de recherche et d'analyse conçu pour analyser et indexer des volumes de données importantes www.elastic.co/fr/elasticsearch

Enfin, il est aussi envisagé de mettre en place de nouvelles manières de rechercher de l’information notamment par thématique (tag) pour faciliter la navigation dans la documentation. Nous souhaitons également intégrer un éditeur wysiwyg externe pour permettre au pôle studio de rédiger des documentations avec une visualisation en temps réel, et ainsi faciliter la prise en main de l'outil par des rédacteurs non techniques.

En conclusion, ces perspectives d'évolution permettront de continuer à améliorer cette solution, en gardant en tête la réponse aux besoins de l’entreprise et des collaborateurs tout en facilitant l'accès à l'information au sein de l’entreprise.

Conclusion

La réalisation de ce mémoire sur la gestion de la documentation technique au sein de l'entreprise a été une expérience riche en apprentissages. En effet, le sujet s'est avéré très intéressant autant sur le plan des recherches et de l'analyse des problématiques du secteur d'activité. L'élaboration de cette solution m'a permis de prendre du recul et de la hauteur sur les besoins et les objectifs d'une *TPE* du domaine de l'informatique.

La mise en place de cet outil est un élément essentiel permettant de fluidifier le transfert des connaissances et des savoirs-faires entre les collaborateurs de *blue2i*. Les résultats obtenus avec cette solution ont été très positifs et ont démontré l'importance de la gestion de la documentation pour le bon fonctionnement de l'entreprise. Les collaborateurs de l'entreprise sont globalement tous satisfaits de la nouvelle solution et n'ont pas rencontré de difficultés majeures à la prise en main de cet outil. La qualité et l'accessibilité des informations ont été grandement améliorées, ce qui permet de gagner du temps au quotidien. Les mises à jour de documentation sont désormais plus fluide, car au plus proche du code, tout en étant contrôlé car soumises à une revue de code systématique pour bénéficier de retours constructifs pour une amélioration continue. Enfin, l'ensemble des procédures techniques sont désormais documentées dès les premières étapes de recherche sur un nouvel outil, ce qui permet de ne perdre aucune valeur à l'éventuel départ d'un salarié. Cependant, la mise en œuvre de ce projet à connu certaines difficultés comme le manque de temps sur certaines phases comme les tests, dû à la nécessité de jongler entre diverses tâches sur différents projets.

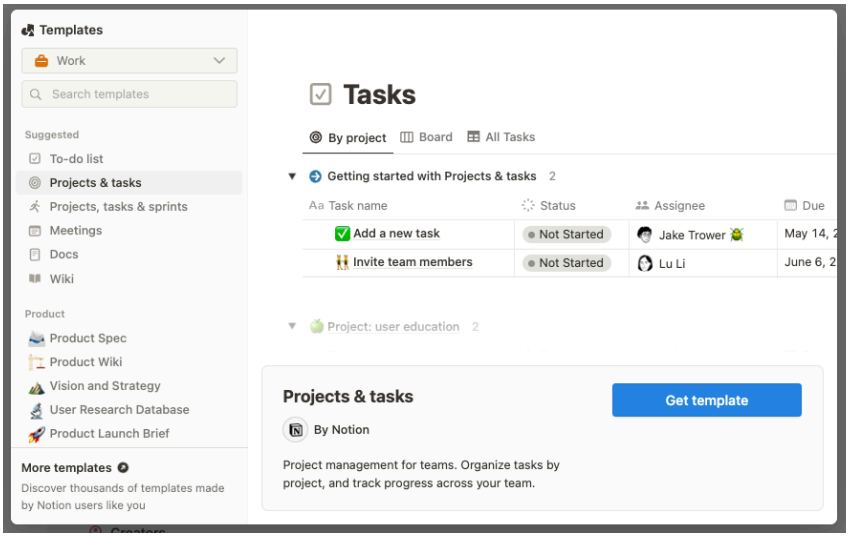
En ce qui me concerne, ce projet a été une réelle opportunité de développement personnel et professionnel. J'ai pu mettre en pratique mes compétences techniques et méthodologiques, mais aussi développer mon sens de l'écoute et de la communication pour répondre aux besoins et attentes de l'entreprise. La phase de recherche m'a aussi permis de prendre du recul sur la réflexion stratégique et les enjeux du déploiement d'un tel outil pour *blue2i*. Tout au long de ce projet, j'ai pu développer mes compétences en gestion de projet et en rédaction technique, mais aussi découvrir de nouveaux outils et méthodes de travail. Je suis fier d'avoir pu apporter ma pierre à l'édifice de *blue2i* et d'avoir contribuer à l'amélioration continue de l'entreprise pour l'aider à atteindre ses objectifs.

En conclusion, la mise en œuvre d'une solution de gestion de documentation technique est un projet complexe mais essentiel pour une entreprise dans le domaine de l'informatique. Les résultats obtenus grâce à la mise en place de la solution sont très satisfaisants et je suis convaincu que celle-ci apportera une réelle valeur ajoutée à *blue2i* sur le long terme.

Annexes

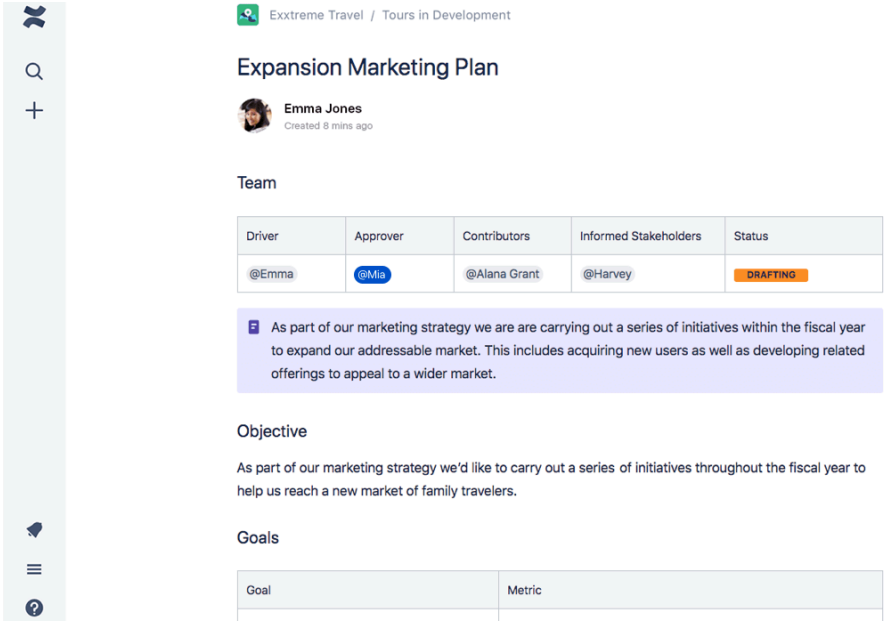
1 - Captures d'écran des différentes solutions de wiki

Notion :



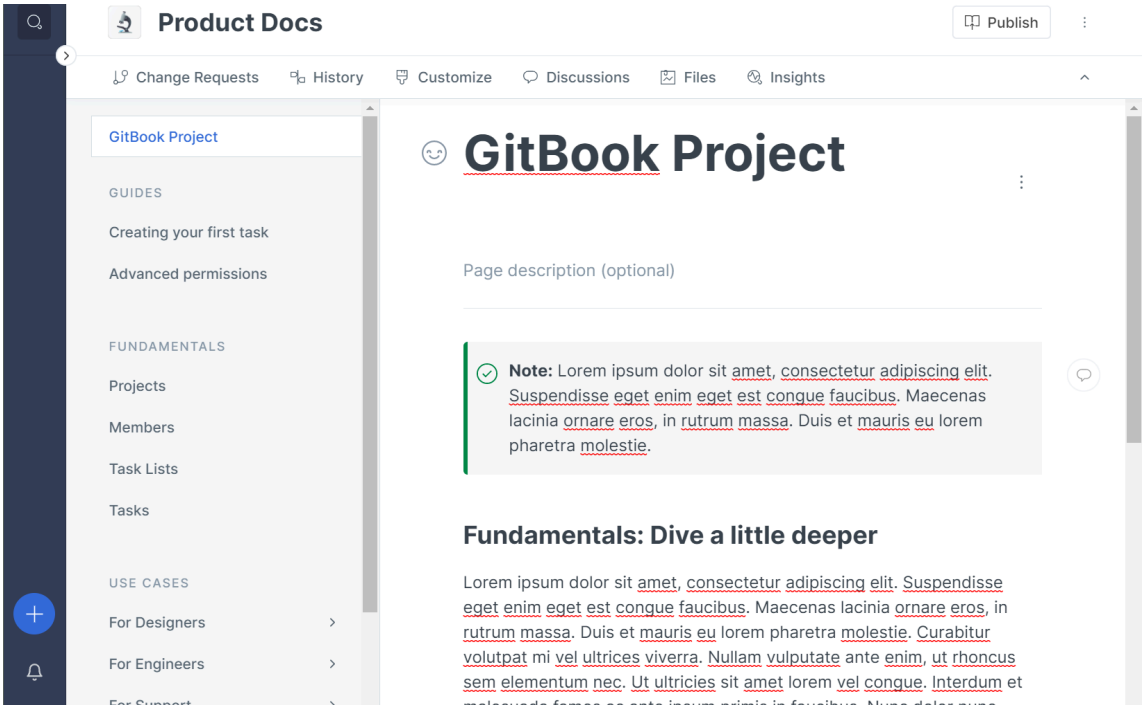
Notion, *The ultimate guide to Notion templates*, www.notion.so/help/guides/the-ultimate-guide-to-notion-templates

Confluence :



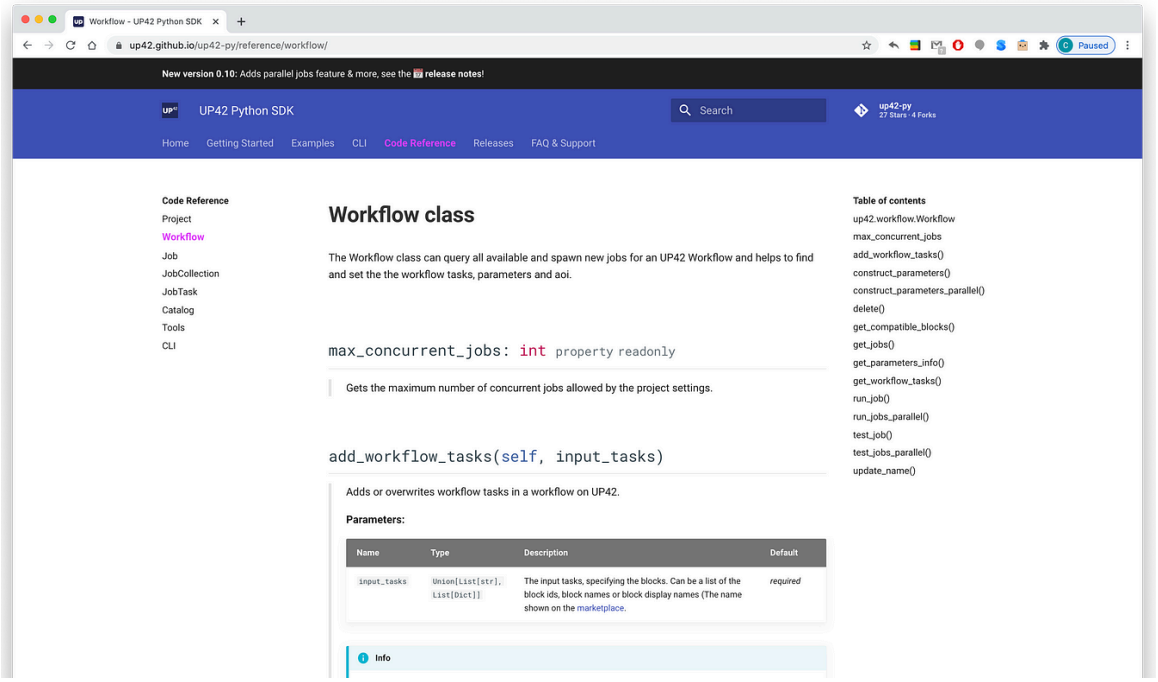
BDM, *Confluence : Un outil de travail collaboratif complet et bon marché*, www.blogdumoderateur.com/tools/confluence/

Gitbook :



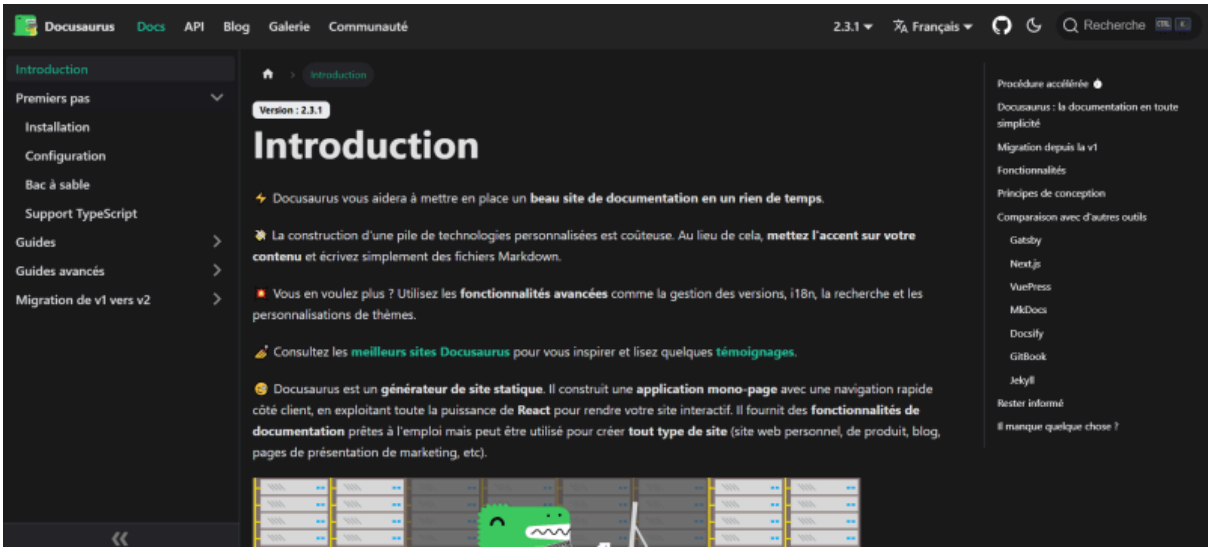
Rohith Namala, *GitBook: An Overview of the Online Platform for Writing and Publishing Books and Documentation*, 26 Mars 2023, www.linkedin.com/pulse/gitbook-overview-online-platform-writing-publishing-books-namala/

Mkdocs :



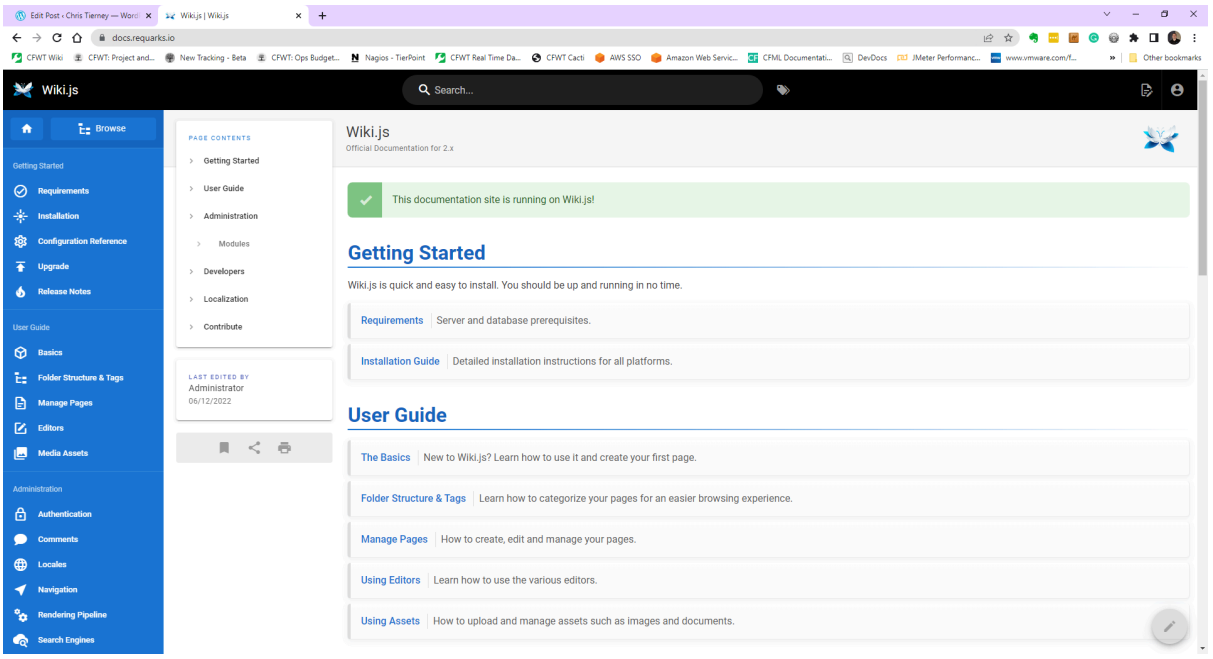
Christoph Rieke, *Documenting a Python package with mkdocs-material, 22 Août 2020,* *chrieke.medium.com/documenting-a-python-package-with-code-reference-via-mkdocs-material-b4a45197f95b*

Docusaurus :



Maxime Guilbert, *Docusaurus - Ou la manière la plus simple de créer votre documentation technique,* 20 Mars 2023, dev.to/mxglt/docusaurus-ou-la-maniere-la-plus-simple-de-creer-votre-documentation-technique-26gj

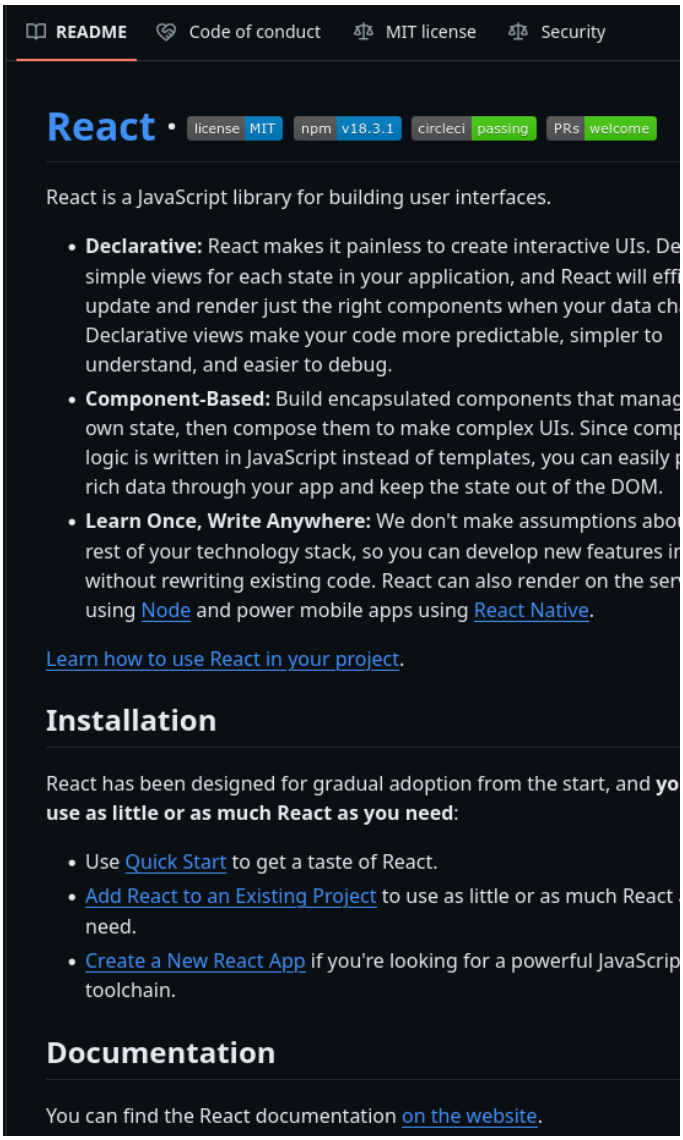
Wiki.js :

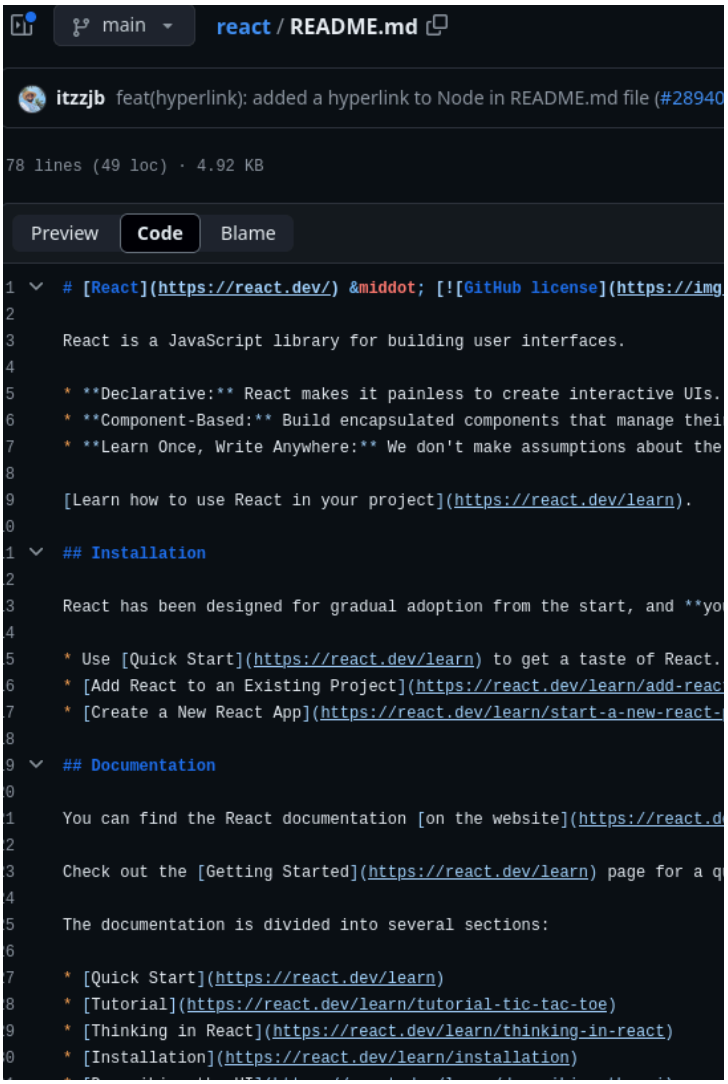


Chris Thierney, *Wiki.js Active Directory Authentication Configuration*, 19 Juin 2022, christierney.com/2022/07/18/wiki-js-active-directory-authentication-configuration/

2 - Exemple de plateforme intégrant la syntaxe markdown nativement

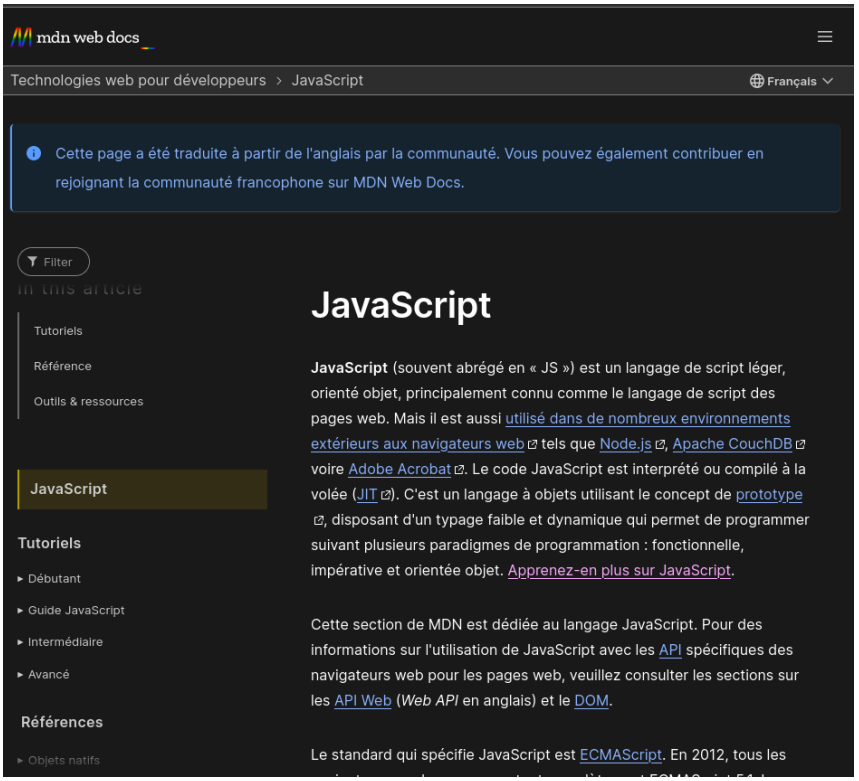
Github



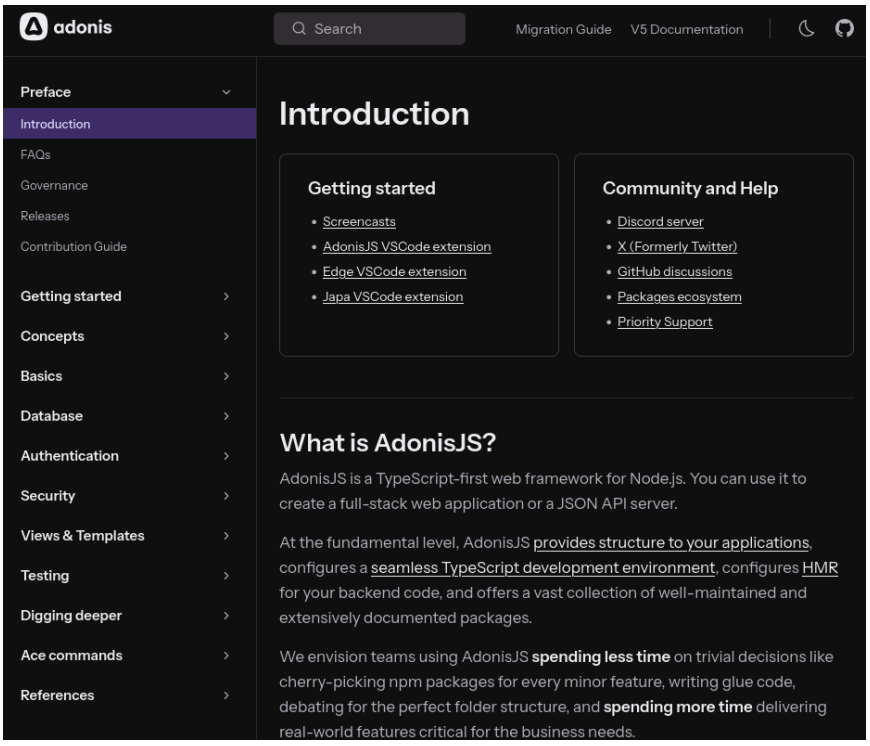


React (open-source project) on Github, github.com/facebook/react

3 - Exemples de documentations construite à partir du format Markdown

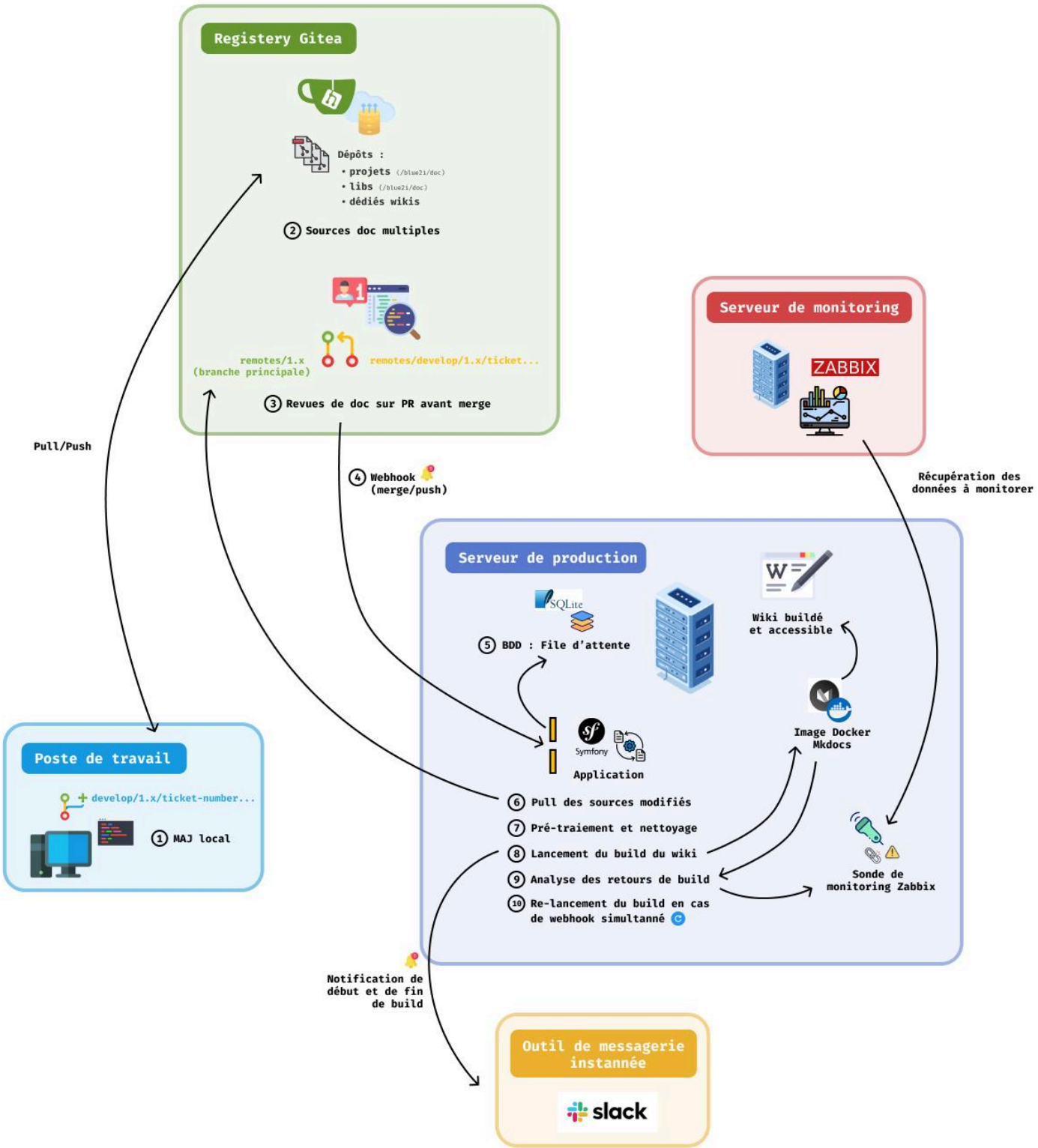


JavaScript, MDN web docs, developer.mozilla.org/fr/docs/Web/JavaScript



AdonisJS, docs.adonisjs.com/guides/preface/introduction

4 - Schéma d'architecture de la solution mise en oeuvre



5 - Budget estimatif de la mise en oeuvre de la solution

Phase	Tâche	Durée estimative	Durée de la phase
Conception	Veille technique et tests de différents outils	21h	59h
	Validation des choix techniques	7h	
	Définition des tâches et objectifs d'étapes	10h	
	Définition de l'architecture logiciel	7h	
	Tests initiaux et validation du POC	14h	
Production	Mise à jour locale des sources de documentation	7h	69h
	Agrégation des sources multiples	7h	
	Mise en place des webhooks de push sur les dépôts sources	3h	
	Pré-traitement et nettoyage des sources de documentation	21h	
	Build du wiki final depuis les sources	14h	
	Analyse des retours de build et réaction en conséquence	10h	
	Mise en place des sondes de monitoring	7h	
Livraison	Déploiement de la solution	7h	30h
	Analyse des premiers retours utilisateurs et ajustements	14h	
	Réunion de clôture, identification et planification des perspectives d'évolution	4h	
	Formation de maintenance en interne	5h	
Maintenance (mensuel)	Correction de bugs	2h	5h
	Mise à jour des outils	3h	
Total estimé			153h

6 - Spécifications fonctionnelles

Introduction

Ce document décrit les spécifications fonctionnelles de l'application de gestion de documentation **b2iwiki**. L'application permet de gérer les sources de documentation en Markdown et de construire un wiki statique à partir de ces sources.

Mise à jour locale des sources de documentation

Les développeurs effectuent leurs modifications sur leurs machines locales et modifient les sources de documentation sur la branche correspondant à leur ticket, comme *develop/1.x/ticket-number-dénomination-branche*.

Sources de documentation multiples

Différents dépôts git sont utilisés pour stocker les sources de documentation, tels que les dépôts de code de projet, de librairies et de documentations spécifiques. Les dépôts spécifiques consacrés à la documentation sont classés selon leur type, comme les procédures, les connaissances ou les formations. En revanche, dans les dépôts de code, la documentation doit être présente dans le dossier */blue2i/doc* à la racine du dépôt.

Contrôle par revue de code

Une fois les modifications apportées, le code doit être validé par l'intermédiaire d'une demande de pull-request. La demande d'intégration est examinée par le CTO ou le responsable du dépôt, qui fournit des requêtes de modification si besoin. Après la validation des modifications, la branche de travail est fusionnée avec la branche principale.

Webhooks (notifications) de push sur les dépôts

Après avoir accepté et fusionné la PR (pull request) dans la branche principale, Gitea envoie une notification par l'intermédiaire d'un webhook (événements de push et merge) sur une route d'une application Symfony avec les informations du dépôt modifié.

File d'attente

Dans une base de données SQLite, l'application Symfony gère une file d'attente des dépôts à mettre à jour. Il est possible de lancer un seul processus de build en même temps. Au début du processus une notification Slack indiquant le début du build est lancée sur un channel dédié : "#builds".

Récupération des sources modifiées

Les dépôts ciblés par le webhook sont mis à jour vers leur dernière version sur le serveur de production.

Pré-traitement et nettoyage des sources de documentation

Le code source est nettoyé pour y supprimer toute trace du code et organisé pour ne contenir que la documentation avec une architecture normalisée pour le futur build avec MkDocs.

Build du wiki final depuis les sources avec l'image docker de MkDocs

Le wiki final est construit à partir de sources de documentation préalablement préparées grâce à l'image docker *squidfunk/mkdocs-material* et docker-compose pour la mise en œuvre.

Analyse des retours de build

Ensuite, on analyse le retour de construction du wiki afin de repérer les anomalies de construction tels que les liens internes cassés ou les anomalies bloquantes. Les anomalies sont ensuite transmises à la sonde de monitoring pour faire remonter l'anomalie au responsable du projet dans Zabbix (outil de monitoring).

Notification de mise à jour terminée

Ensuite, la référence de modification du dépôt de la base SQLite est supprimée et une notification Slack de fin de build est envoyée.

Re-lancement du processus en cas de webhook simultanés

Après la fin du build, la base de données SQLite est vérifiée et le processus est relancé si des notifications de mise à jour ont été envoyées pendant la durée du build.

7 - Spécifications techniques

Architecture

La solution de gestion de documentation est composée de plusieurs éléments :

- Des dépôts git hébergés sur un serveur Gitea pour stocker les sources de documentation.
- Une application Symfony développée en PHP 8 pour gérer les notifications de mise à jour, la file d'attente et le build du wiki.
- Un serveur de production sous Linux (Ubuntu2204) pour héberger l'application Symfony et le wiki.
- Une base de données SQLite pour stocker les informations de la file d'attente sur le serveur de production.
- L'outil de monitoring Zabbix (installé sur un autre serveur) pour surveiller l'état de l'application.

Dépôts git sur Gitea

Les dépôts git sont organisés de la manière suivante :

- Les dépôts de code de projet et de librairies contiennent un dossier */blue2i/doc* à la racine du dépôt pour stocker les sources de documentation.
- Les dépôts dédiés à la documentation sont organisés en fonction de leur forme : procédures, connaissances ou formations.

Application Symfony

L'application Symfony a pour rôle de :

- Récupérer les notifications de mise à jour de Gitea via des webhooks.
- Gérer une file d'attente des dépôts à mettre à jour dans une base de données SQLite.
- Lancer le processus de build du wiki pour chaque dépôt de la file d'attente.
- Envoyer des notifications Slack en début et en fin de processus de build.
- Analyser les retours de build pour détecter les anomalies et les envoyer à l'outil de monitoring Zabbix.

Serveur de production

Le serveur de production sous Ubuntu2204 permet d'héberger l'application Symfony et le wiki et donc de lancer le processus de build et les éventuels scripts annexes.

Base de données SQLite

La base de données SQLite est utilisée pour stocker les informations de la file d'attente gérée par l'application Symfony.

Outil de monitoring Zabbix

L'outil de monitoring Zabbix permet de surveiller l'état de la solution. Il est configuré pour récupérer les anomalies envoyées par l'application Symfony et pour envoyer des notifications aux responsables en cas de problème.

8 - Questionnaire de satisfaction de la mise en place de la nouvelle solution

1 - À quel point êtes-vous satisfait de la nouvelle solution de gestion de documentation ?

- Très satisfait
- Satisfait
- Neutre
- Insatisfait
- Très insatisfait

Résultats :

- Salarié 1 : Satisfait
- Salarié 2 : Très satisfait
- Salarié 3 : Satisfait
- Salarié 4 : Satisfait

2 - Avez-vous rencontré des difficultés lors de l'utilisation de la nouvelle solution de gestion de documentation ?

- Oui
- Non

Résultats :

- Salarié 1 : Non
- Salarié 2 : Non
- Salarié 3 : Oui
- Salarié 4 : Non

3 - Si vous avez répondu "oui" à la question précédente, pouvez-vous décrire brièvement les difficultés rencontrées ?

- Réponse du Salarié 3 : J'ai eu du mal à comprendre comment m'organiser dans la rédaction depuis une page blanche.

4 - Dans l'ensemble, pensez-vous que la nouvelle solution de gestion de documentation a amélioré la qualité et l'accessibilité de la documentation dans l'entreprise ?

- Oui
- Non

Résultats :

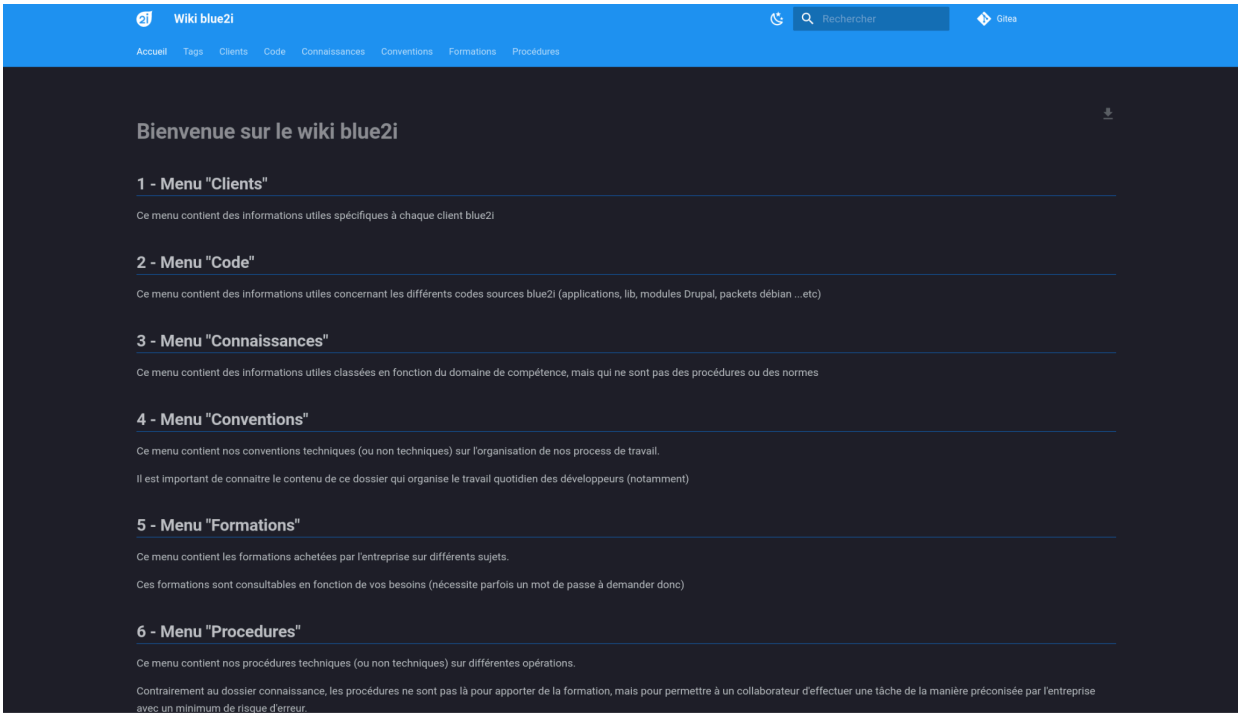
- Salarié 1 : Oui
- Salarié 2 : Oui
- Salarié 3 : Oui
- Salarié 4 : Oui

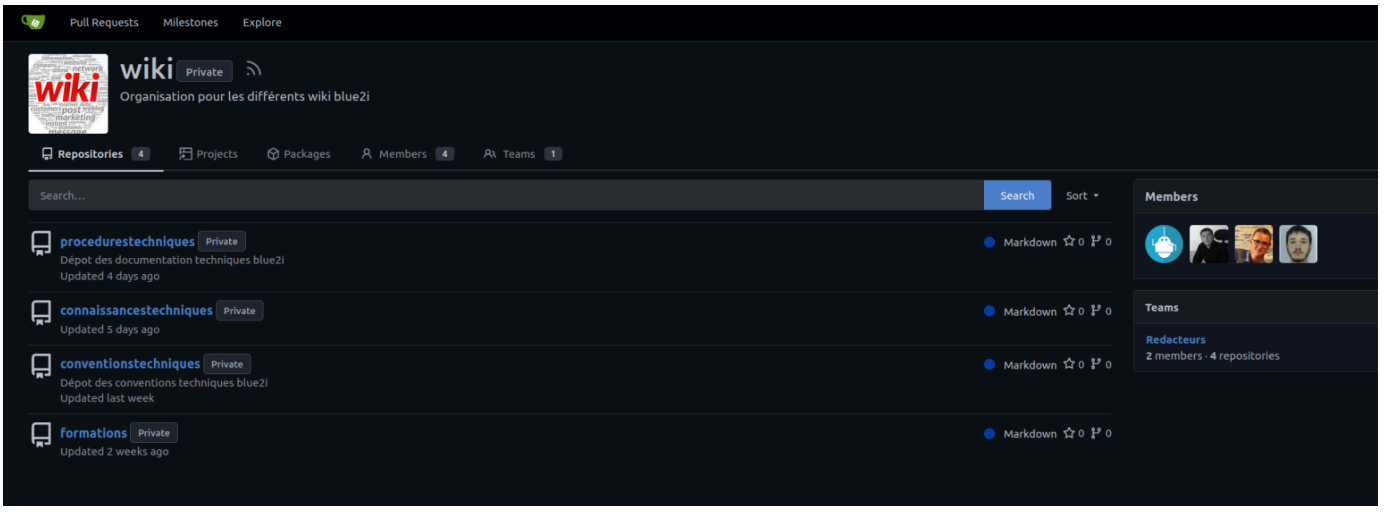
5 - Avez-vous des suggestions ou des commentaires supplémentaires concernant la nouvelle solution de gestion de documentation ?

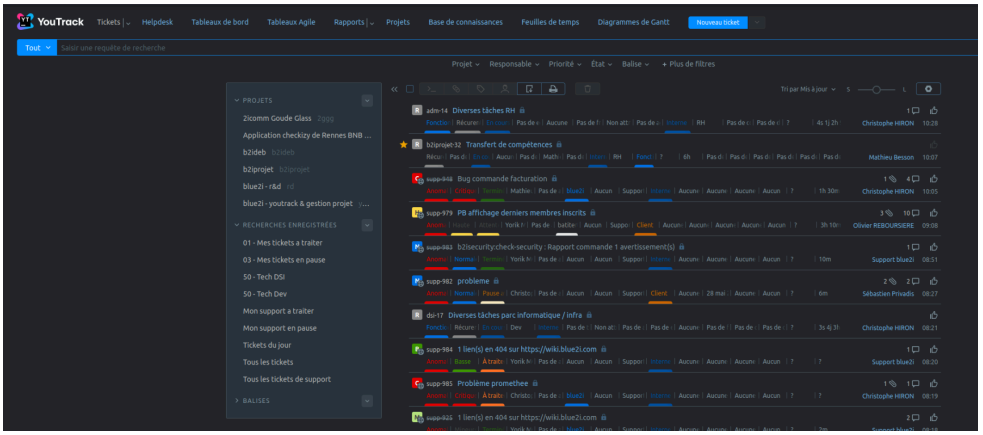
Réponses :

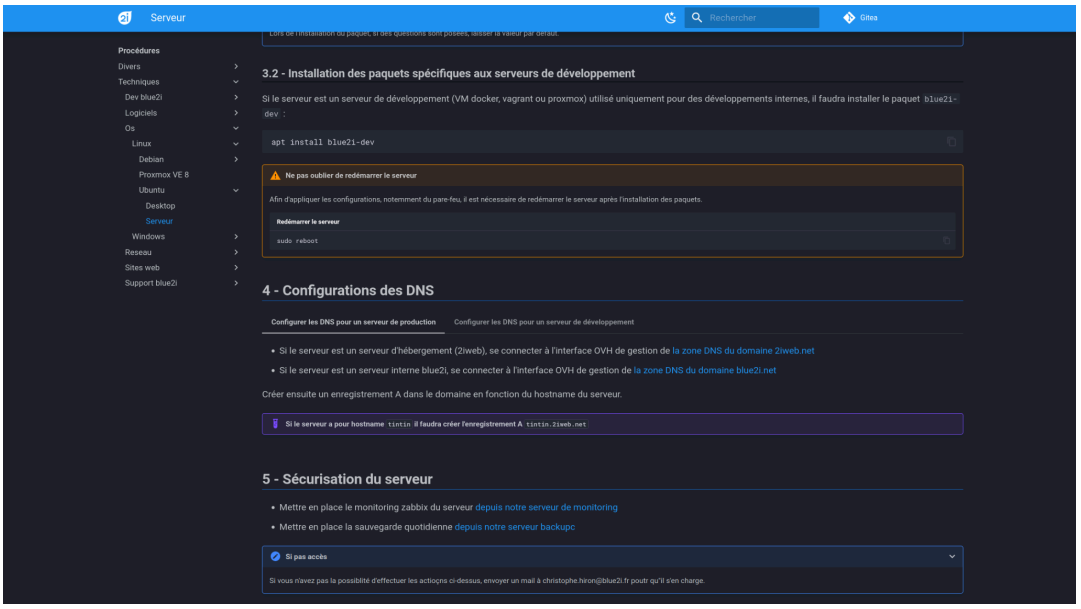
- Salarié 1 : Le wiki est facile à utiliser et permet de trouver rapidement des documents sans solliciter mes collègues.
- Salarié 2 : La nouvelle solution est efficace. Elle a amélioré la qualité et l'accessibilité de la documentation dans l'entreprise.
- Salarié 3 : La nouvelle solution est bien mais j’ai eu du mal à comprendre tout le mécanisme. J'ai, je pense, besoin d’un peu de temps pour m'y habituer.
- Salarié 4 : Je suis très satisfait de la nouvelle solution. Elle est facile à utiliser et elle permet de gagner du temps dans la recherche de doc.

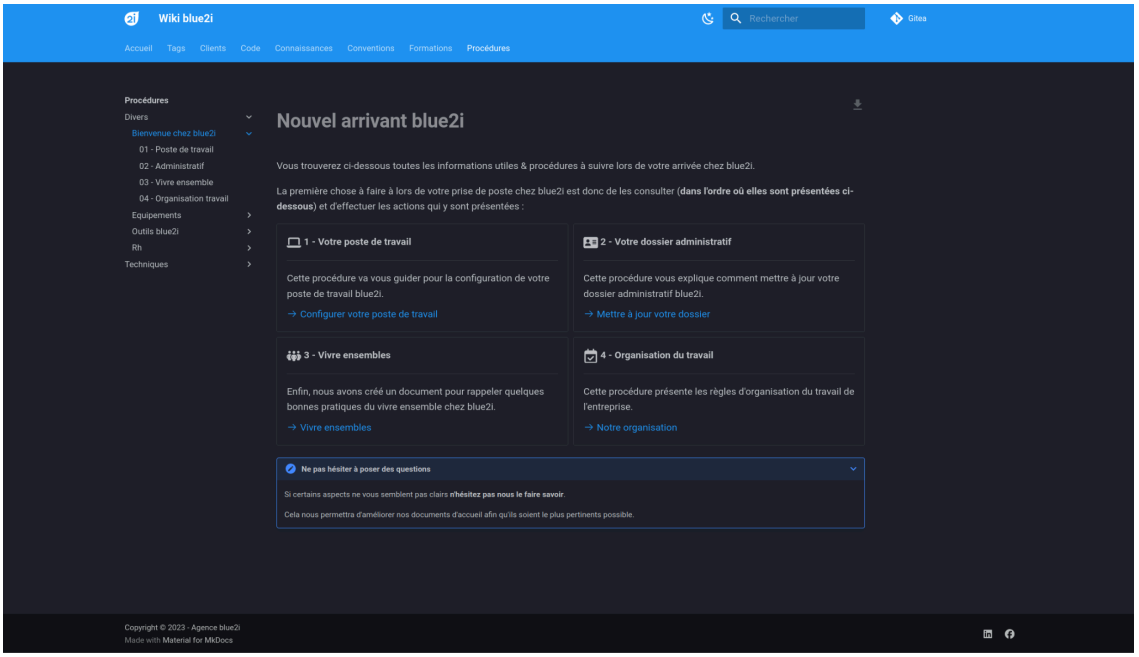
9 - Captures d'écran du wiki, de Gitea et de Youtrack

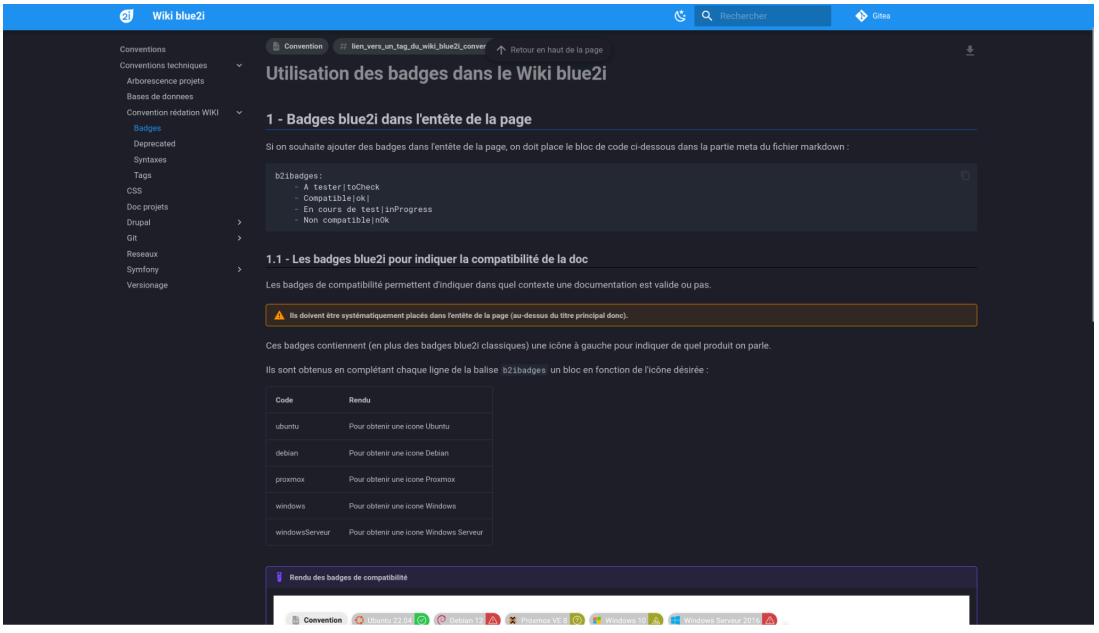


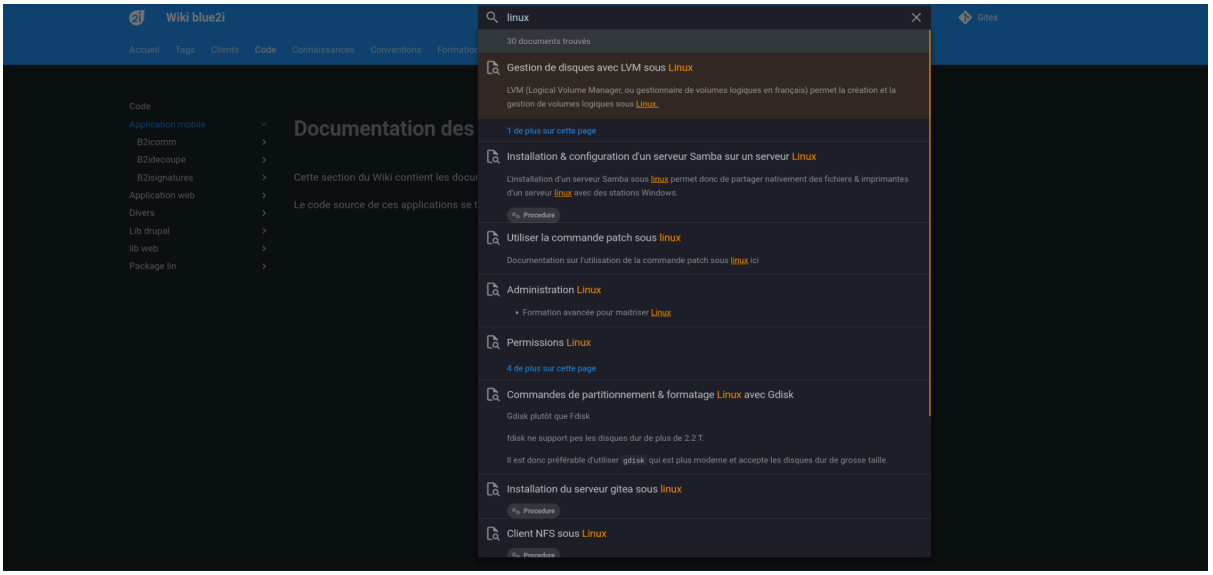


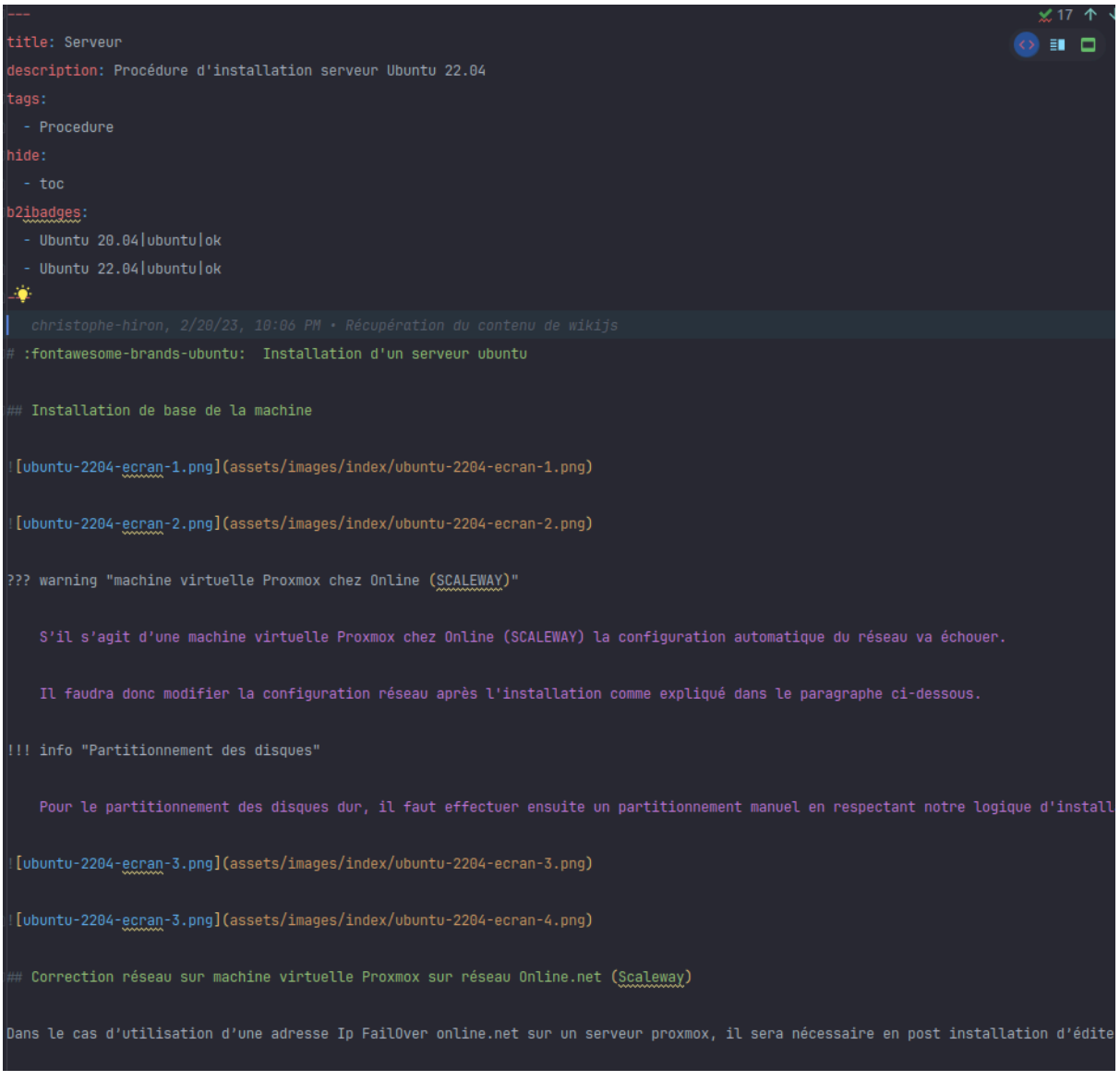



Serveur
Rechercher
Gitea
Procédures
Divers
Techniques
Dev blue2i
Logiciels
Os
Linux
Debian
Proxmox VE 8
Ubuntu
Desktop
Serveur
Windows
Reseau
Sites web
Support blue2i
3.2 - Installation des paquets spécifiques aux serveurs de développement
Si le serveur est un serveur de développement (VM docker, vagrant ou proxmox) utilisé uniquement pour des développements internes, il faudra installer le paquet blue2i-dev :
apt install blue2i-dev
Ne pas oublier de redémarrer le serveur
Afin d'appliquer les configurations, notemment du pare-feu, il est nécessaire de redémarrer le serveur après l'installation des paquets.
Redémarrer le serveur
sudo reboot
4 - Configurations des DNS
Configurer les DNS pour un serveur de production
Configurer les DNS pour un serveur de développement
Si le serveur est un serveur d'hébergement (2iweb), se connecter à l'interface OVH de gestion de la zone DNS du domaine 2iweb.net
Si le serveur est un serveur interne blue2i, se connecter à l'interface OVH de gestion de la zone DNS du domaine blue2i.net
Créer ensuite un enregistrement A dans le domaine en fonction du hostname du serveur.
Si le serveur a pour hostname tintin il faudra créer l'enregistrement A tintin.2iweb.net
5 - Sécurisation du serveur
Mettre en place le monitoring zabbix du serveur depuis notre serveur de monitoring
Mettre en place la sauvegarde quotidienne depuis notre serveur backupc
Si pas accès
Si vous n'avez pas la possiblité d'effectuer les actioçns ci-dessus, envoyer un mail à christophe.hiron@blue2i.fr poutr qu'il s'en charge.


Wiki blue2i
Rechercher
Gitea
Accueil
Tags
Clients
Code
Connaissances
Conventions
Formations
Procédures
Procédures
Divers
Bienvenue chez blue2i
01 - Poste de travail
02 - Administratif
03 - Vivre ensemble
04 - Organisation travail
Equipements
Outils blue2i
Rh
Techniques
Nouvel arrivant blue2i
Vous trouverez ci-dessous toutes les informations utiles & procédures à suivre lors de votre arrivée chez blue2i.
La première chose à faire à lors de votre prise de poste chez blue2i est donc de les consulter (dans l'ordre où elles sont présentées ci-dessous) et d'effectuer les actions qui y sont présentées :
1 - Votre poste de travail
Cette procédure va vous guider pour la configuration de votre poste de travail blue2i.
→ Configurer votre poste de travail
2 - Votre dossier administratif
Cette procédure vous explique comment mettre à jour votre dossier administratif blue2i.
→ Mettre à jour votre dossier
3 - Vivre ensembles
Enfin, nous avons créé un document pour rappeler quelques bonnes pratiques du vivre ensemble chez blue2i.
→ Vivre ensembles
4 - Organisation du travail
Cette procédure présente les règles d'organisation du travail de l'entreprise.
→ Notre organisation
Ne pas hésiter à poser des questions
Si certains aspects ne vous semblent pas clairs n'hésitez pas nous le faire savoir.
Cela nous permettra d'améliorer nos documents d'accueil afin qu'ils soient le plus pertinents possible.
Copyright © 2023 - Agence blue2i
Made with Material for MkDocs


Wiki blue2i
Rechercher
Gitea
Conventions
Conventions techniques
Arborescence projets
Bases de donnees
Convention rédation WIKI
Badges
Deprecated
Syntaxes
Tags
CSS
Doc projets
Drupal
Git
Reseaux
Symfony
Versionage
Convention
lien_vers_un_tag_du_wiki_blue2i_conver
Retour en haut de la page
Utilisation des badges dans le Wiki blue2i
1 - Badges blue2i dans l'entête de la page
Si on souhaite ajouter des badges dans l'entête de la page, on doit place le bloc de code ci-dessous dans la partie meta du fichier markdown :
b2ibadges:
- A tester|toCheck
- Compatible|ok|
- En cours de test|inProgress
- Non compatible|nOk
1.1 - Les badges blue2i pour indiquer la compatibilité de la doc
Les badges de compatibilité permettent d'indiquer dans quel contexte une documentation est valide ou pas.
Ils doivent être systématiquement placés dans l'entête de la page (au-dessus du titre principal donc).
Ces badges contiennent (en plus des badges blue2i classiques) une icône à gauche pour indiquer de quel produit on parle.
Ils sont obtenus en complétant chaque ligne de la balise b2ibadges un bloc en fonction de l'icône désirée :
Code
Rendu
ubuntu
Pour obtenir une icone Ubuntu
debian
Pour obtenir une icone Debian
proxmox
Pour obtenir une icone Proxmox
windows
Pour obtenir une icone Windows
windowsServeur
Pour obtenir une icone Windows Serveur
Rendu des badges de compatibilité
Convention
Ubuntu 22.04
Debian 12
Proxmox VE 8
Windows 10
Windows Serveur 2016


Wiki blue2i
Accueil
Tags
Clients
Code
Connaissances
Conventions
Gitea
Code
Application mobile
B2icomm
B2idecoupe
B2isignatures
Application web
Divers
Lib drupal
lib web
Package lin
Documentation des
Cette section du Wiki contient les docu
Le code source de ces applications se t
linux
30 documents trouvés
Gestion de disques avec LVM sous Linux
LVM (Logical Volume Manager, ou gestionnaire de volumes logiques en français) permet la création et la gestion de volumes logiques sous Linux.
1 de plus sur cette page
Installation & configuration d'un serveur Samba sur un serveur Linux
L'installation d'un serveur Samba sous linux permet donc de partager nativement des fichiers & imprimantes d'un serveur linux avec des stations Windows.
Procedure
Utiliser la commande patch sous linux
Documentation sur l'utilisation de la commande patch sous linux ici
Administration Linux
Formation avancée pour maitriser Linux
Permissions Linux
4 de plus sur cette page
Commandes de partitionnement & formatage Linux avec Gdisk
Gdisk plutôt que Fdisk
fdisk ne support pes les disques dur de plus de 2.2 T.
Il est donc préférable d'utiliser gdisk qui est plus moderne et accepte les disques dur de grosse taille.
Installation du serveur gitea sous linux
Procedure
Client NFS sous Linux
Procedure


---
title: Serveur
description: Procédure d'installation serveur Ubuntu 22.04
tags:
- Procedure
hide:
- toc
b2ibadges:
- Ubuntu 20.04|ubuntu|ok
- Ubuntu 22.04|ubuntu|ok
christophe-hiron, 2/20/23, 10:06 PM • Récupération du contenu de wikijs
:fontawesome-brands-ubuntu: Installation d'un serveur Ubuntu
Installation de base de la machine
![ubuntu-2204-ecran-1.png](assets/images/index/ubuntu-2204-ecran-1.png)
![ubuntu-2204-ecran-2.png](assets/images/index/ubuntu-2204-ecran-2.png)
??? warning "machine virtuelle Proxmox chez Online (SCALEWAY)"
S'il s'agit d'une machine virtuelle Proxmox chez Online (SCALEWAY) la configuration automatique du réseau va échouer.
Il faudra donc modifier la configuration réseau après l'installation comme expliqué dans le paragraphe ci-dessous.
!!! info "Partitionnement des disques"
Pour le partitionnement des disques dur, il faut effectuer ensuite un partitionnement manuel en respectant notre logique d'install
![ubuntu-2204-ecran-3.png](assets/images/index/ubuntu-2204-ecran-3.png)
![ubuntu-2204-ecran-3.png](assets/images/index/ubuntu-2204-ecran-4.png)
Correction réseau sur machine virtuelle Proxmox sur réseau Online.net (Scaleway)
Dans le cas d'utilisation d'une adresse Ip FailOver online.net sur un serveur proxmox, il sera nécessaire en post installation d'édite

10 - Grille d'évaluation de la pratique professionnelle

Grille d'évaluation de la pratique professionnelle

Cette fiche est à inclure dans les annexes du mémoire

Candidat : BESSON MATHIEU

Entreprise : BLUE2i

Tuteur : CHRISTOPHE HIRON

Critères	Non acquis 0	Partiellement acquis 1	Acquis 2	Dépassé 3	Total
Savoir-faire					/ 60
Atteinte des objectifs			X		2
Méthodes d'organisation			X		2
Utilisation des procédures internes				X	3
Capacité à travailler en équipe			X		2
Prise d'initiatives				X	3
Rigueur				X	3
Sens des responsabilités				X	3
Capacité de prise de recul			X		2
Maîtrise de l'écrit			X		2
Expression orale			X		2
Aptitude à mener à bien une mission			X		2
Aptitude à gérer son temps			X		2
Capacité à identifier ses priorités			X		2
Aptitude à intégrer des contraintes			X		2
Notion de hiérarchie			X		2
Notion d'objectif individuel et collectif			X		2
Identification des problèmes potentiels			X		2
Aptitude à manager un groupe de travail		X			1
Aptitude au contact téléphonique			X		2
Aptitude au relationnel face à face			X		2

Savoir-être					/ 30
Conscience professionnelle				X	3
Dynamisme				X	3
Sociabilité			X		2
Ponctualité / assiduité			X		2
Maturité			X		2
Ténacité			X		2
Réactivité			X		2
Autonomie d'apprentissage				X	3
Intérêt porté à l'entreprise				X	3
Adaptabilité à la culture d'entreprise				X	3
				Total	/ 90
				Total	/ 20

Appréciation du tuteur professionnel :

Très bonne expérience pour l'entreprise.
Mathieu s'est montré impliqué et intéressé par toutes les missions qu'il a eut a traiter durant son alternance.
Il a fait preuve d'autonomie, rigueur, initiative.
Il a de très bonnes aptitudes techniques et méthodologiques et nous aurions souhaité qu'il puisse rester dans l'entreprise.
Bonne chance pour la suite de ton parcours professionnel.

Date 31/05/2024

Signature et cachet de l'entreprise :

C. HIRON

blue2i AGENCE DIGITALE
36 rue Blaise Pascal
35220 Châteaubourg
02 23 37 35 69
511 536 799 RCS Rennes
APE 6201Z
SARL au capital de 8000 €
www.blue2i.com

Bibliographie

Works Cited

Ahmed, Kamran. “driver.js.” *driver.js*, driverjs.com.

Allen, Robert. “Swagger-PHP.” *Annotations*, zircote.github.io/swagger-php/guide/annotations.html.

Altassian. “Adieu les silos, bonjour le travail d'équipe.” *Altassian*, www.atlassian.com/fr/software/confluence.

Ambysoft Inc. “Just Barely Good Enough (JBGE) Artifacts: An Agile Core Practice.” *Agile Modeling*, http://agilemodeling.com/essays/barelygoodenough.htm.

Barbara, Gil. “React Joyride. Create guided tours for your apps.” *React Joyride. Create guided tours for your apps*, https://react-joyride.com/.

Beck, Kent, et al. “Manifesto for Agile Software Development.” 2001, http://agilemanifesto.org/.

Commonmark, and John MacFarlane. “Markdown Flavors.” *Github*, www.github.com/commonmark/commonmark-spec/wiki/markdown-flavors.

Cunningham, Ward. “Wikitexte.” *Wikipédia*, https://fr.wikipedia.org/wiki/Wikitexte.

Donath, Martin. “Documentation that simply works.” *Material for MkDocs*, squidfunk.github.io/mkdocs-material.

Eclipse Foundation, and AsciiDoc. “AsciiDoc.” *AsciiDoc*, https://asciidoc.org/.

Elasticsearch B.V. “Elasticsearch : Le moteur de recherche et d'analyse distribué officiel | Elastic.” *Elasticsearch : Le moteur de recherche et d'analyse distribué officiel | Elastic*, www.elastic.co/fr/elasticsearch.

Eurecia. “Le logiciel RH pour une expérience épanouissante au travail.” *Le logiciel RH pour une expérience épanouissante au travail*, https://www.eurecia.com/.

Facebook. “Build optimized websites quickly, focus on your content.” *Build optimized websites quickly, focus on your content*, https://www.docusaurus.io/.

Garousi, G., et al. *Usage and usefulness of technical software documentation: An industrial case study*. Belfast, Information and Software Technology, Queen's University Belfast, 2015, https://pureadmin.qub.ac.uk/ws/files/180209101/IST2013_Special_Issue_July30.pdf.

GitBook. “Effortless documentation starts here.” *GitBook – Knowledge management for technical teams*, www.gitbook.com.

Github. “GitHub Flavored Markdown Spec.” *GitHub Flavored Markdown Spec*, github.github.com/gfm.

Gitlab. “Gitlab.” *Gitlab*, https://about.gitlab.com/.

Go Gitea. “Private, Fast, Reliable DevOps Platform.” *Gitea Official Website*, https://about.gitea.com/.

Google. “Technical Writing Courses (Google).” https://developers.google.com/tech-writing.

Groupe APICIL. “Communiqué de presse : Qualité de vie au travail : un salarié désengagé coûte 14 310€ par an à son entreprise, selon l’IBET 2020.” *Groupe APICIL*, 16 Octobre 2020, https://www.groupe-apicil.com/wp-content/uploads/2020/10/161020_CP_APICIL_IBET2020.pdf.

Gruber, John. “Markdown.” 2004, https://daringfireball.net/projects/markdown/.

HEC (Hautes Etudes Commerciales). “Nouvelles perspectives pour réduire l’impact du turnover dans l’informatique.” *www.hec.fr*, http://www.hec.fr/Knowledge/Strategie-et-Management/Management-des-Ressources-Humaines/Nouvelles-perspectives-pour-reduire-l-impact-du-turnover-dans-l-informatique.

Jekyll. “Transform your plain text into static websites and blogs.” *Transform your plain text into static websites and blogs.*, jekyllrb.com.

Jetbrains. “Gestion de projet performante pour toutes les équipes de votre organisation.” *YouTrack : gestion de projet pour toutes les équipes de votre organisation*, https://www.jetbrains.com/fr-fr/youtrack/.

Jones, Katherine, and Deloitte Consulting LLP. “Onboarding Software Solutions 2014, Bersin by Deloitte.” 2014, https://s3.eu-central-1.amazonaws.com/maison-moderne-paperjam-gms-prod/files/101514-onboarding-final.pdf.

Koskimies, Sakri. “phpDocumentor - Generate GitHub/GitLab-Ready Markdown Documentation.” *Github*, https://github.com/Saggre/phpDocumentor-markdown.

Legrand, Charlotte, and Welcome To The Jungle. “ombien coûte vraiment un mauvais recrutement ?” *Welcome To The Jungle*, 12 Avril 2021, https://www.welcometothejungle.com/fr/articles/cout-mauvais-recrutement.

MacFarlane, John. “CommonMark Spec.” *CommonMark Spec*, https://spec.commonmark.org/.

MacFarlane, John. “Pandoc a universal document converter.” *Pandoc a universal document converter*, www.pandoc.org.

Mäkiranta, Eero. *Piloting markdown-bases documentation*. Tempere, Tempere University, 2023, https://trepo.tuni.fi/bitstream/handle/10024/146213/MakirantaEero.pdf?sequence=2.

Martin, Robert C. *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall, 2008.

McBreen, Pete. *Software Craftsmanship: The New Imperative*. Addison-Wesley Professional, 2001.

McFeely, Shane, et al. “This Fixable Problem Costs U.S. Businesses $1 Trillion.” *Gallup*, 13 Mars 2019, https://www.gallup.com/workplace/236051/real-value-getting-exit-interview-right.aspx.

Microsoft. “Visual Studio Code.” *Visual Studio Code*, code.visualstudio.com.

Microsoft, and Github. “Github.” *Github*, https://github.com/.

MUNCI (Mouvement pour une Union Nationale des Consultants en Informatique). *Consultation public pour l'élaboration du plan France numérique 2020*. Paris, 2011. *www.economie.gouv.fr*,

https://www.economie.gouv.fr/files/files/import/2011_france_numerique_consultation/2011_france_numerique_munci.pdf.

Narçon, Laetitia. “Cycle en V vs Agile : le match !” *Findle - Conseil en transformation digitale & supply chain du Retail*, https://findle.fr/cycle-v-vs-cycle-agile/.

Nextcloud GmbH. “Reprenez le contrôle de vos données.” *Nextcloud - Online collaboration platform*, nextcloud.com/fr/.

Nirav, Kanani. “Github Pre-commit Hook Setup In Ruby On Rails for maintaining coding standards and productive.” *Dev Community*, 2022, https://dev.to/kanani_nirav/github-pre-commit-hook-setup-in-ruby-on-rails-12m3.

Notion. “Write, plan, organize, play.” *Write, plan, organize, play*, www.notion.so.

OASIS DITA TC. “DITA Open Toolkit.” *DITA Open Toolkit*, https://www.dita-ot.org/.

Olmstead, Levi. “9 Types of In-App Training to Enable Users (+Examples).” *Whatfix*, 2024, https://whatfix.com/blog/in-app-training/.

OpenAPI Initiative. “The world's most widely used API description standard.” *The world's most widely used API description standard*, OpenAPI Initiative.

Panopto. “Le partage inefficace des connaissances coûte 47 millions de dollars par an aux grandes entreprises.” *Panopto*, 31 Juillet 2020, https://www.panopto.com/fr/company/news/inefficient-knowledge-sharing-costs-large-businesses-47-million-per-year/.

Patard, Alexandra. “Bilan et perspectives de l’emploi dans la tech en 2023.” *blogdumoderateur.com*, 9 Janvier 2023, https://www.blogdumoderateur.com/bilan-perspectives-emploi-it-2023-michael-page-tech/. Accessed 15 Mai 2024.

phpDocumentor. “Because code and documentation are meant to be together.” *phpDocumentor*, https://www.phpdoc.org/.

PHP-FIG. “PHP Standards Recommendations.” *PHP-FIG — PHP Framework Interop Group - PHP-FIG*, https://www.php-fig.org/psr/.

PHPStan. “PHPStan.” *PHPStan*, www.phpstan.org.

"PhpStorm : l'IDE PHP, par JetBrains." *Make Your Software Vision a Reality*, www.jetbrains.com/fr-fr/phpstorm.

PICART, Claude, and INSEE (Institut national de la statistique et des études économiques). "Une rotation de la main d'œuvre presque quintuplée en 30 ans plus qu'un essor des formes particulières d'emploi, un profond changement de leur usage." *www.insee.fr*, https://www.insee.fr/fr/statistiques/fichier/1381150/F1402.pdf.

Plenik, Piotr. "Clean Code PHP." *Github*, https://github.com/piotrplenik/clean-code-php.

Procida, Daniele. "Diátaxis." https://diataxis.fr/.

Python-Markdown. "Python-Markdown." *Python-Markdown*, python-markdown.github.io.

Randstad. "Le coût réel du turnover en entreprise." *Randstad*, 17 Mai 2023, https://www.randstad.fr/recruteurs/magazine-instant-rh/competitivite-performance/cout-reel-turnover-entreprise/.

Slack, and Microsoft. "Made for people. Built for productivity." *Made for people. Built for productivity.*, slack.com.

SmartBear Software. "API Development forEveryone." *API Development forEveryone*, https://swagger.io/.

SmartBear Software. "Swagger Petstore." *Swagger Petstore*, https://petstore.swagger.io/.

Sommerville, Ian. *Software Documentation*. Lancaster, Lancaster University, http://www.literateprogramming.com/documentation.pdf.

the Sphinx developers. "Sphinx documentation." *Sphinx documentation*, www.sphinx-doc.org.

SQLite Consortium. "SQLite." *SQLite*, sqlite.org/.

Stripe. "Markdoc powers Stripe documentation." *Markdoc powers Stripe documentation*, markdoc.dev.

TeamPasswordManager, and Ferran Barba. "The Team Password Manager." *The Team Password Manager*, https://teampasswordmanager.com/.

The Textile Language Development Team. "Textile Markup Language Documentation." *Textile Markup Language Documentation*, https://textile-lang.com/.

Wiki.js. “The most powerful and extensible open source Wiki softwar.” *Wiki.js*, www.js.wiki.

Wikipedia. “Diagramme de séquence.” *Wikipedia*, https://fr.wikipedia.org/wiki/Diagramme_de_s%C3%A9quence.

Write the Docs. “Write the Docs.” *Welcome to our community! — Write the Docs*, https://www.writethedocs.org/.

Zabbix LLC. “Zabbix :: The Enterprise-Class Open Source Network Monitoring Solution.” *Zabbix :: The Enterprise-Class Open Source Network Monitoring Solution*, www.zabbix.com.