

Dossier de projet professionnel



KRAV MAGA SPIRIT

Refonte du site web du club Krav Maga Spirit

—

Titre préparé : RNCP 37873 - Concepteur développeur
d'applications

Introduction au projet.....	4
Présentation personnelle.....	4
Présentation du projet.....	4
Document de projet.....	5
Historique des révisions.....	5
Contexte de réalisation.....	6
Objet du document.....	7
Présentation de l'entreprise.....	8
Entreprise.....	8
Organisation.....	9
Le Projet.....	11
Cadrage et présentation.....	11
Le projet.....	11
Méthodologie de travail.....	12
Résultats souhaités.....	14
Exclusions et portée du projet.....	15
Contraintes.....	16
Tolérances.....	17
Parties prenantes.....	18
Plan de communication.....	19
L'équipe projet.....	21
Organigramme.....	21
Ressources du projet.....	22
RACI du projet.....	24
Analyse des risques.....	26
Evaluation des risque.....	26
Plan d'action.....	29
Efficience.....	30
Planification.....	31
Conception et réalisation :.....	33
Présentation de l'existant.....	33
Choix des technologie et justification.....	35

Conception de la base de données.....	36
Architecture de l'application.....	38
Présentation des outils.....	42
Conception.....	45
Présentation de la réalisation.....	52
Organisation du déploiement.....	56
Tests et recette.....	57
Bilan projet :	58
Analyse Budgétaire.....	58
Résultat atteints.....	59
Bilan personnel :	60
Retours d'expériences.....	60
Projet professionnel.....	61
Annexe :	62
Documentation.....	62
Graphiques.....	62
Base de données.....	63
Diagrammes.....	63
Autres.....	64
Glossaire	65

Introduction au projet

Présentation personnelle

Bonjour, je m'appelle Corentin Roussel, j'ai 28 ans. Je suis actuellement en troisième année de Bachelor Spécialisation Logiciel à La Plateforme. Je suis en alternance dans l'entreprise Cityway, que je présenterai plus bas. J'ai découvert la programmation grâce à la formation "Start", proposée aussi par La Plateforme, qui proposait une découverte du hardware, des réseaux et du développement web, où j'ai validé mon diplôme RNCP de niveau 5 Développeur d'application web et web mobile. Aujourd'hui, je vais donc vous présenter mon dossier de projet pour le diplôme RNCP de niveau 6 TP - Concepteur Développeur d'Applications.

Présentation du projet

Le projet que je vais vous présenter est un site web réalisé en groupe de 3 : avec l'aide de Fabien Ricca et Léa Dubois.

KMS ou **Krav Maga Spirit** est une association sportive basée à Marseille dédiée, comme son nom l'indique, à la pratique du krav maga. Le projet provient d'une demande d'un client réel, le but étant de faire une refonte d'un site vitrine avec un design simple et épuré. L'ancien site, étant devenu obsolète, ne permettait plus d'être maintenu correctement. Les gérants du club ne pouvaient tout simplement plus mettre à jour les informations mises à disposition des visiteurs du site, ce qui créait un problème pour les personnes souhaitant se renseigner sur ce même club.

Le but du projet est donc de remettre au goût du jour aussi bien la partie front-end, pour que les clients qui arrivent sur le site puissent trouver les informations nécessaires plus facilement, qu'un back-end qui va permettre aux gérants du club de pouvoir personnaliser plus facilement le site web. Cela va, à la fin, améliorer la qualité de vie de chacune des parties utilisant le nouveau site.

Les compétences demandées et couvertes par le projet m'ont permis de découvrir, d'améliorer ou d'approfondir mes connaissances, que ce soit sur l'analyse des besoins d'un client réel, la conception architecturale, la rédaction d'un cahier des charges ou la gestion de projet — des points que l'on voit rarement sur des projets d'école.

Avec un back-end qui permet une personnalisation plus complète afin de rendre son utilisation plus simple pour le principal utilisateur. Le projet sera donc déployé sur un VPS (Virtual private Server et un est une machine qui héberge tous les logiciels et les données nécessaires au fonctionnement d'une application ou d'un site web) OVH, avec un système de routing Traefik. Le front et le back sont dockerisés. Les technologies utilisées sont Angular en front, Node.js en back, et les tests sont réalisés avec Karma et Jasmine en front, Jest et Prisma en back.

Document de projet

Historique des révisions

Date de révision	Résumé des changements	Changement marqué
01/06/2026	Mis en place du sommaire du document	Ajout de sections
02/07/2025	Démarrage de la rédaction du dossier	Changement suivi
04/07/2025	Changement de la structure	Réécriture importante
09/07/2025	Continuation de la rédaction	Changement suivi
15/07/2025	Continuation de la rédaction	Changement suivi
19/07/2025	Continuation de la rédaction et relecture	Changement suivi
21/07/2025	Envoi à mon équipe pour relecture	Changement suivi
24/07/2025	Applications des corrections apportés par mon tuteur	Réécriture importante
30/07/2025	Finalisation du dossier	Changement suivi
01/08/2025	Rendu du dossier	RAS

Contexte de réalisation

Afin de pouvoir valider mon année de Bachelor 3 Spécialisation Logiciel dispensée à La Plateforme, je dois réaliser, avec un groupe de trois personnes, un projet de groupe à choix libre d'une durée de 6 mois, qui jongle entre alternance et école. L'objectif de ce projet est de permettre de développer un projet qui englobe toutes les compétences nécessaires pour valider le diplôme.

Les lacunes principales de l'ancien site web se situent en grande partie au niveau du back-end : un manque de personnalisation (ajout de photos, galeries, modification d'instructeurs, téléchargement de fichiers), ce qu'il n'était pas possible de faire.

Du côté public, le site était très vieillissant, avec une navigation difficile. Les informations omniprésentes rendaient l'expérience confuse et l'utilisateur était rapidement perdu en arrivant sur les pages.

L'objectif de la refonte du site KMS est donc d'analyser les besoins du client afin de créer un site web qui réponde aux attentes de l'administrateur (en matière de personnalisation), et ensuite d'apporter notre expertise sur la partie publique pour offrir à l'utilisateur une expérience fluide et agréable.

Objet du document

Ce mémoire a pour objectif de présenter toute la réflexion menée durant ce projet, les objectifs qui ont été fixés, et surtout les attentes initiales confrontées aux résultats réels obtenus à la fin de celui-ci.

Nous verrons donc, dans une première partie, une présentation de mon entreprise et de son organisation.

Dans une seconde partie, le projet sera expliqué plus en détail : seront présentés le projet, l'équipe impliquée, les expressions des besoins fonctionnels, l'analyse des risques, ainsi que la planification.

Ensuite seront analysées la conception et la réalisation de la solution, qu'il s'agisse de l'existant, du choix des technologies, de la conception de la base de données et de la solution, des outils, de la réalisation et du déploiement.

Puis, un premier bilan du projet sera mis en avant, portant sur l'analyse budgétaire et les résultats obtenus.

Enfin, un bilan personnel sera présenté, basé sur le retour d'expérience du projet. Dans les dernières pages se trouveront les annexes, référencées tout au long du document, ainsi qu'un glossaire récapitulant les notions importantes et les sigles utilisés.

Présentation de l'entreprise

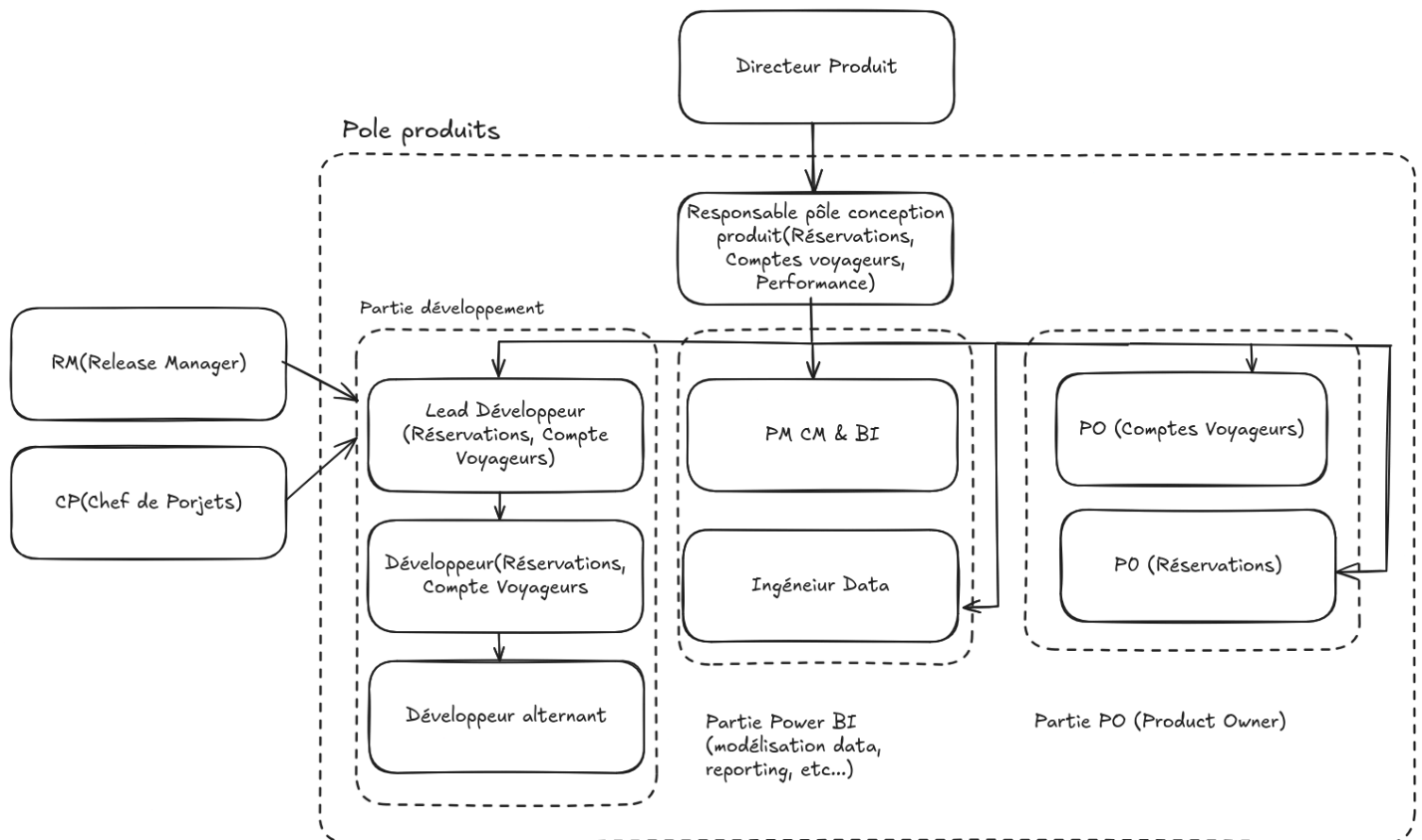
Entreprise

Cityway, filiale du groupe Transdev, est une entreprise basée à Aix-en-Provence, spécialisée dans les solutions numériques de mobilité intelligente. Depuis plus de 20 ans, elle conçoit, développe et déploie des plateformes MaaS (Mobility as a Service) pour les collectivités, les opérateurs de transport et les citoyens. Présente dans plus de 150 villes en France et à l'international, Cityway propose des applications web et mobiles intégrant l'information voyageur en temps réel, le calcul d'itinéraires multimodaux, la réservation, l'achat de titres et la gestion des trajets.

Le pôle Produits dont je fais partie à Cityway couvre plusieurs briques clés du MaaS, notamment le Transport à la Demande (TAD), le compte voyageur, et l'analyse des données via Power BI. Le module TAD permet aux territoires de proposer une offre de transport flexible, accessible par réservation via mobile ou web, avec une gestion intelligente des trajets, des horaires et des tournées. Le compte voyageur centralisé repose sur une architecture d'IAM (Identity and Access Management) intégrant le SSO (Single Sign-On), ce qui permet à l'utilisateur d'accéder à tous les services (réservation, paiements, abonnements) de manière fluide et sécurisée. Pour le pilotage, les équipes utilisent Power BI afin de mettre en forme et analyser les données issues des usagers : suivi de fréquentation, performance des lignes, activité du TAD, indicateurs clés, etc. Ces tableaux de bord facilitent la prise de décision et permettent aux clients de mieux comprendre et optimiser leurs réseaux de transport.

Enfin, Cityway se distingue par une culture d'entreprise orientée vers l'innovation, l'agilité et la transition écologique. En plus de ses activités techniques, elle accompagne les territoires dans leur transformation numérique via des offres d'incubation, des expérimentations de nouvelles mobilités, et une expertise reconnue sur l'ensemble de la chaîne de valeur du transport public. Sa mission : rendre la mobilité plus simple, connectée et inclusive pour tous.

Organisation



1. Direction du pôle Produits

Au sommet de l'organigramme, le Directeur Produit définit la vision stratégique globale de l'ensemble des produits numériques de l'entreprise. Il pilote l'orientation à long terme, arbitre les grandes décisions et veille à la cohérence entre les pôles produits, les besoins des clients (autorités organisatrices de mobilité) et les enjeux d'innovation du marché MaaS.

Directement en lien avec lui, le responsable du pôle conception produit joue un rôle opérationnel et transversal central. Il supervise le pôle dédié aux modules « Réservations », « Comptes Voyageurs » et « Performance », en assurant la coordination entre les différentes équipes (développement, PM, data) et en garantissant la cohérence fonctionnelle des produits. Il anime les rituels produits (ateliers de cadrage, revues, roadmaps), s'assure de la qualité de l'expérience utilisateur et priorise les évolutions en lien avec les retours clients, les contraintes techniques et les objectifs stratégiques. Il est à la croisée des chemins entre le métier, la technique et la donnée.

2. Partie développement technique

Le Lead Développeur est chargé de l'encadrement technique de l'équipe développement. Il pilote les choix d'architecture, la qualité du code, la revue technique et l'intégration continue. Son périmètre couvre les produits Réservations et Comptes Voyageurs, assurant que chaque livraison est stable, sécurisée et fonctionnelle.

Sous sa responsabilité se trouvent :

- des Développeurs qui implémentent les évolutions des fonctionnalités (réservation TAD, gestion d'identité IAM, etc.), corrigent les anomalies et participent à la documentation technique ;
- un Développeur alternant, en montée en compétences, qui contribue activement aux projets.

L'équipe est également accompagnée par :

- le Release Manager (RM), responsable de la mise en production, du déploiement et du suivi des versions logicielles ;
- le Chef de Projets (CP), qui assure le pilotage opérationnel : planning, relations client, jalons contractuels, coordination inter-équipes.

Cette structure garantit une approche fluide entre conception produit et développement, avec un fort niveau de communication entre les rôles.

3. Partie produit (Project Management)

Les Product Managers (PM) sont garants de la vision fonctionnelle des modules :

- Le PM Réservations pilote les fonctionnalités liées au Transport à la Demande (création de créneaux, règles d'éligibilité, logiques de trajets, etc.) et veille à la simplicité de l'expérience de réservation.
- Le Product Manager Comptes Voyageurs gère la partie identité numérique de l'utilisateur : création de compte, authentification (SSO/IAM), gestion de profils, données personnelles.

Ils traduisent les besoins utilisateurs ou clients en spécifications fonctionnelles claires et exploitables, collaborent avec les développeurs et la data pour garantir une vision produit cohérente, et priorisent les évolutions dans les sprints agiles. Leur objectif : garantir que le produit réponde réellement aux usages, avec une forte valeur ajoutée métier.

4. Partie PowerBI/Données (Project Management)

L'équipe Power BI a pour mission de valoriser les données issues des services MaaS proposés. Elle se compose de :

- Un Product Manager CM & BI, responsable des besoins fonctionnels liés à la visualisation et à l'analyse des données. Il conçoit les tableaux de bord (statistiques d'usage, taux de réservation, comportements voyageurs), échange avec les clients internes/externes, et s'assure que les reporting soient clairs et pertinents.
- Un Ingénieur Data, qui développe la modélisation des données, les flux de traitement, l'intégration dans Power BI, et gère les performances des requêtes et la qualité des données restituées.

Cette synergie permet un pilotage en temps réel des performances du produit, une meilleure prise de décision, et une amélioration continue basée sur des données fiables et visuellement accessibles.

Le Projet

Cadrage et présentation

Le projet

Le projet **KMS** a pour ambition de moderniser en profondeur la présence numérique du club **Krav Maga Spirit**, en concevant une plateforme web à la fois moderne, fonctionnelle et adaptée aux usages actuels. Le site existant souffre aujourd'hui de plusieurs limitations majeures : une navigation peu intuitive, une interface visuelle dépassée et un espace d'administration impraticable, freinant totalement la mise à jour des contenus par les responsables.

La solution proposée repose sur la création d'une plateforme structurée en deux espaces complémentaires.

L'**espace public** offrira aux visiteurs une présentation claire et attractive du club, intégrant toutes les informations pratiques essentielles (horaires, lieux, recommandations), un aperçu de l'histoire et des valeurs du Krav Maga, une galerie photo, une présentation de l'équipe encadrante, ainsi qu'un accès simplifié aux documents administratifs et pédagogiques (programmes de ceinture, formulaires d'inscription, etc.). Un formulaire de contact permettra également d'initier facilement un échange avec les gérants du club.

En parallèle, un **espace administratif** sera mis à disposition des gestionnaires du club. Conçu pour être simple et intuitif, cet espace permettra de gérer efficacement

les contenus du site : ajout, modification ou suppression de galeries photo, gestion des profils de l'équipe pédagogique, mise à jour des documents importants. Cette interface repensée a pour objectif de faciliter l'autonomie des responsables tout en réduisant le temps consacré à la maintenance du site.

À travers cette refonte, le projet **KMS** vise à améliorer significativement la communication du club, à renforcer sa visibilité en ligne, et à offrir un outil de gestion simple et efficace. À moyen terme, des évolutions sont envisagées, telles que l'intégration d'un bandeau d'actualités, la centralisation de la communication par e-mail avec les adhérents, ou encore la mise en place d'un véritable système de gestion des membres.

En résumé, le projet **KMS** a pour finalité de doter **Krav Maga Spirit** d'un outil numérique performant et ergonomique, servant à la fois de vitrine valorisante pour le public et de plateforme de gestion optimisée pour les administrateurs, contribuant ainsi à la modernisation et au rayonnement du club.

Méthodologie de travail

Pour mener à bien le projet **KMS**, notre équipe a adopté une méthodologie hybride, inspirée des principes agiles (notamment **Scrum**) et adaptée aux contraintes spécifiques de notre formation en alternance. Cette approche nous a permis d'assurer un développement progressif et structuré, tout en restant flexibles face aux imprévus ou aux ajustements nécessaires.

Approche générale

Le projet a été découpé en **sprints hebdomadaires**, correspondant à nos semaines de présence à l'école. Chaque sprint débutait par une **réunion de planification**, suivie d'un **développement intensif** sur la semaine, et se clôturait par un **point de bilan**. Cette organisation cyclique a permis de maintenir un rythme soutenu, en équilibrant travail en autonomie et collaboration lors des regroupements.

En dehors des temps en présentiel, des contributions volontaires étaient encouragées, notamment durant les périodes en entreprise.

Organisation de l'équipe

L'équipe projet s'est structurée autour des compétences de chacun, tout en favorisant la polyvalence :

- **Léa Dubois** : Développement du front-end public et contribution à la conception générale.
- **Corentin Roussel** : Développeur principal du back-end et référent technique.

- **Fabien Ricca** : En charge du front-end de l'interface d'administration, participation à la conception globale, et responsable de la CI/CD (intégration et déploiement continus).

Cette répartition a favorisé une bonne complémentarité et une vision transversale du projet, soutenue par une collaboration active à chaque étape.

Outils et gestion de projet

La gestion du projet s'est appuyée sur l'utilisation de GitLab (issues board) pour le suivi des tâches, la répartition du travail et la communication technique. Chaque fonctionnalité était associée à un ticket, donnant lieu à la création d'une branche dédiée dans le dépôt Git.

Deux réunions rythment chaque sprint :

- **Lundi matin** : planification des tâches et définition des priorités.
- **Vendredi soir** : récapitulatif des avancées, retours d'expérience et préparation du sprint suivant.

Une communication continue via nos outils collaboratifs a permis de maintenir la cohésion de l'équipe et de résoudre rapidement les obstacles.

Processus de développement

Le développement s'est appuyé sur les bonnes pratiques du travail collaboratif :

- Création de branches spécifiques par fonctionnalité.
- Intégration continue grâce à des merges réguliers.
- Tests systématiques avant chaque fusion de code : aucun merge n'était autorisé si les tests échouent.
- Revues de code croisées pour garantir la qualité, faciliter la montée en compétence et garder une vision globale du projet.

Communication avec le client

La collaboration avec les responsables du club Krav Maga Spirit s'est faite à travers des étapes de validation clés : présentation des maquettes, démonstrations fonctionnelles et retours sur l'ergonomie. Ces échanges ont permis d'ajuster rapidement nos choix techniques et graphiques en fonction des besoins réels du club.

Phases du projet

Le projet s'est structuré en quatre grandes étapes :

1. **Conception** : Analyse des besoins, architecture technique, maquettes et définition des fonctionnalités.

2. **Développement** : Répartition des tâches, implémentation et construction des interfaces.
3. **Tests** : Vérifications techniques, tests d'intégration et retours utilisateurs.
4. **Déploiement** : Mise en production sur l'infrastructure OVH.

Résultats souhaités

Le projet de refonte du site web du club Krav Maga Spirit vise à aboutir à une plateforme moderne, performante et parfaitement adaptée aux besoins du club et de ses utilisateurs. Plusieurs résultats clés ont été définis pour mesurer la réussite du projet :

- **Un site web moderne et fiable** : La solution livrée devra permettre de présenter les activités du club de manière claire, attractive et à jour, tout en permettant une gestion autonome du contenu par les administrateurs.
- **Une interface intuitive et responsive** : Que ce soit pour le grand public, l'interface devra offrir une navigation fluide, ergonomique et adaptée à tous types de supports (smartphone, tablette, ordinateur).
- **Une administration simplifiée** : Le back-office devra faciliter considérablement la gestion du contenu (photos, documents, équipes...), sans nécessiter de compétences techniques avancées.
- **Une meilleure visibilité en ligne** : Le nouveau site devra renforcer l'image et la présence numérique du club, avec l'objectif d'attirer de nouveaux adhérents et de valoriser ses valeurs et son ambiance.
- **Une documentation claire** : Des guides techniques et utilisateurs accompagneront la plateforme afin d'assurer une prise en main rapide et une maintenance facilitée.
- **Des performances optimisées** : Le site devra être fluide et rapide, être stable et supporter une forte fréquentation sans dégradation de l'expérience utilisateur.
- **Un gain de temps pour les gestionnaires** : L'interface d'administration devra alléger et simplifier les tâches quotidiennes liées à la communication et à la mise à jour du site.
- **La satisfaction des utilisateurs** : Membres, visiteurs et administrateurs doivent trouver dans la solution une expérience agréable, efficace et alignée avec leurs attentes.
- **Une plateforme évolutive** : L'architecture du site devra permettre l'intégration future de nouvelles fonctionnalités (comme de l'actualités ou des outils de communication internes).

En résumé, ce projet a pour ambition de livrer une solution web à la fois efficace, durable et évolutive, répondant aux besoins actuels du club tout en anticipant ses futurs développements.

Exclusions et portée du projet

Étant sur un projet avec un client réel nous avons un besoin de déterminer la portée du projet et d'exclure les parties qui n'allaient pas pouvoir être implémentés pour pouvoir respecter les deadlines.

Ce tableau nous permet donc de déterminer le périmètres que l'on peut mettre en place pour pouvoir mener à bien le projet sans être dépassé par des demandes clients qui prendrait du temps et ralentissait la sortie du produit v1

Élément	Inclus dans la portée	Exclus du projet
Fonctionnalités web	Création d'un site web moderne avec espace public et interface d'administration.	Application mobile native.
Gestion de contenu	Mise à disposition d'une interface d'administration pour gérer les contenus (photos, documents, équipe).	Paieement ou inscription en ligne.
Expérience utilisateur	Interface responsive adaptée aux smartphones, tablettes, et ordinateurs.	Accessibilité complète (normes RGAA, multilingue).
Sécurité & accès	Authentification simple pour les administrateurs.	Système avancé de gestion des rôles utilisateurs ou de permissions multiples.
CI/CD	Pipeline simple, GitLab, intégration continue.	Architecture complexe.
Communication	Formulaire de contact intégré.	Messagerie interne ou email automatique groupé.
Documentation	Fourniture d'un guide utilisateur simplifié et documentation technique de base.	Formation des utilisateurs ou accompagnement personnalisé à la prise en main.
Déploiement	Hébergement via OVH, mise en ligne du site, configuration Docker.	Mise en place d'une infrastructure cloud distribuée ou de haute disponibilité.
Évolutivité	Prévue pour ajout futur actu, mailing, gestion membres.	Intégration immédiate de toutes les fonctionnalités futures.

Contraintes

Pour répondre aux exigences du projet de refonte du site web du club Krav Maga Spirit, plusieurs contraintes doivent être prises en compte tout au long de sa réalisation.

- **Contrainte temporelle** : La date de livraison du projet, initialement prévue pour le 1er août 2025, a été reportée au 1er octobre 2025. Cette échéance constitue une date butoir ferme pour la présentation du dossier de projet, ce qui impose un respect strict du planning et une gestion rigoureuse du temps disponible.
- **Contrainte organisationnelle liée à l'alternance** : Le rythme imposé par l'alternance représente une des principales limites du projet. Avec une seule semaine sur quatre passée à l'école, les occasions de travail en équipe sont restreintes. Cela nécessite une organisation précise, une planification détaillée des tâches, ainsi qu'une communication soutenue entre les différentes périodes.
- **Contrainte budgétaire** : Réalisé dans un cadre associatif et bénévole, le projet ne dispose d'aucun financement. Cette situation oblige à utiliser exclusivement des outils gratuits ou open source, à l'exception de certains services comme Amazon S3 pour le stockage, qui peuvent occasionner des frais minimes. Tous les choix technologiques doivent donc respecter cette contrainte financière.
- **Contrainte de ressources humaines** : L'équipe étant composée de trois personnes (Fabien Ricca, Léa Dubois et moi-même), il est nécessaire de répartir efficacement les responsabilités selon les compétences de chacun. Cette contrainte implique une forte polyvalence des membres de l'équipe et une collaboration constante pour assurer la couverture de l'ensemble des besoins du projet.
- **Contrainte de coordination avec le club** : Bien que les membres du club soient disponibles pour les validations, leur engagement bénévole implique une disponibilité variable. Le rythme de validation doit donc s'adapter à leurs contraintes sans compromettre le bon déroulement du projet ni les délais de livraison.
- **Contrainte technologique libre** : L'absence de technologies imposées peut également être considérée comme une contrainte. Elle oblige l'équipe à choisir et justifier des solutions techniques cohérentes, en tenant compte des compétences internes, de la facilité de maintenance et des performances attendues.

Ces différentes contraintes doivent être intégrées dès les premières phases du projet afin d'assurer son bon déroulement, le respect des délais, ainsi que l'adéquation de la solution finale avec les objectifs définis dans un contexte à la fois académique et associatif.

Tolérances

Pour la finalisation du projet KMS des tolérances peuvent être acceptées afin de ne pas bloquer le bon déroulement du projet, mais qui va permettre aussi de respecter nos engagements

Tolérances sur les délais

- **Livraison finale** : un léger décalage peut être toléré si cela permet d'assurer la qualité ou de corriger des bugs bloquants.
- **Livraison intermédiaire** (sprints, maquettes) : tolérance de ± 5 jours selon les contraintes d'alternance ou de disponibilité du client.
- **Délais de validation par le client** : jusqu'à une semaine de délai supplémentaire peut être accepté sans compromettre le planning général.

Tolérances sur le périmètre fonctionnel

- **Aucune flexibilité** n'est accordée sur les fonctionnalités attendues par le client. Le périmètre fonctionnel validé en amont doit être entièrement respecté, sans suppression, modification ou report de fonctionnalité. Chaque module prévu dans le cahier des charges doit être livré et opérationnel.

Tolérances techniques

- **Choix des technologies** : une certaine flexibilité est admise dans le choix des frameworks ou bibliothèques, à condition qu'ils soient open source, bien documentés et maîtrisés par l'équipe.
- **Hébergement** : en cas d'indisponibilité du fournisseur initial, une solution alternative équivalente peut être utilisée, à condition que les coûts restent maîtrisés.

Tolérances budgétaires

- **Frais exceptionnels** : des coûts très faibles (quelques euros) sont acceptables pour des services essentiels, comme l'achat d'un nom de domaine ou un stockage distant.
- **Utilisation de ressources gratuites avec limitations** : tolérée si elle ne compromet pas la performance ou la qualité globale de l'expérience utilisateur.

Tolérances qualité / performance

- **Temps de chargement** : une légère latence sur les pages non critiques (par exemple, la galerie photo) est acceptable, tant que l'expérience utilisateur reste fluide.
- **Interface utilisateur** : certains éléments de design peuvent faire l'objet d'ajustements ou de simplifications si cela permet de respecter les délais de livraison.

Tolérances de documentation

- **Livraison des documents** : le niveau de détail des guides utilisateurs peut être simplifié à condition qu'ils permettent une prise en main rapide.

Parties prenantes

La réussite du projet repose sur l'identification et l'implication des différentes parties prenantes, chacune jouant un rôle spécifique dans la conception, le développement, la validation et l'exploitation de la plateforme web.

1. Le club Krav Maga Spirit (client principal)

- **Rôle** : Commanditaire du projet et bénéficiaire final.
- **Responsabilités** :
 - Définir les besoins fonctionnels et les objectifs.
 - Valider les maquettes, fonctionnalités et livrables.
 - Fournir les contenus nécessaires (textes, photos, documents).
 - Tester la plateforme avant mise en production.
- **Implication** : Essentielle lors des phases de validation et de recueil des besoins.

2. L'équipe projet

Composée de trois membres issus de la formation en alternance :

- **Léa D.** – Développeuse front-end pour la partie publique.
- **Corentin R.** – Développeur back-end principal.
- **Fabien R.** – Développeur front-end de l'interface d'administration, responsable de la CI/CD.

- **Rôle** : Réalisation complète de la solution (conception, développement, tests, déploiement).
- **Responsabilités** :
 - Respect du cahier des charges.
 - Répartition des tâches selon les compétences.
 - Communication régulière avec le client pour valider les étapes du projet.
- **Implication** : Totale durant toutes les phases du projet.

3. L'organisme de formation

- **Rôle** : Encadrement pédagogique du projet dans le cadre de la formation au Titre Professionnel CDA (Concepteur Développeur d'Applications).
- **Responsabilités** :
 - Suivi de l'avancement du projet.
- **Implication** : Modérée, principalement lors des phases de bilan et d'évaluation.

4. Utilisateurs finaux

- **Visiteurs du site** : Personnes intéressées par le club, cherchant des informations ou souhaitant prendre contact.
- **Administrateurs** : Membres du bureau du club chargés de la gestion du contenu.
- **Responsabilités** :
 - Utilisation quotidienne de la plateforme.
 - Retour d'expérience sur l'ergonomie et la fiabilité.
- **Implication** : À partir de la phase de test et lors de l'exploitation.

Plan de communication

1. Objectifs de la communication

- Assurer la coordination entre les membres de l'équipe projet.
- Maintenir un suivi régulier de l'avancement du projet.
- Faciliter la validation des livrables avec le club Krav Maga Spirit.
- Garantir la transmission fluide des informations entre l'équipe, les parties prenantes et les encadrants pédagogiques.

2. Outils et supports utilisés

- **GitLab Issues & Board** : gestion des tâches, backlog, suivi des sprints.

- **GitLab CI/CD** : automatisation des tests, intégration continue et déploiement.
- **Google Chat** : communication quotidienne, échanges techniques entre membres.
- **Email** : échanges formels, envoi de comptes rendus et validations.
- **Documents partagés (Google Docs ou équivalent)** : guides et documentations.
- **Réunions en présentiel** : planification et bilan hebdomadaire lors des semaines en centre.

3. Fréquence et types d'échanges

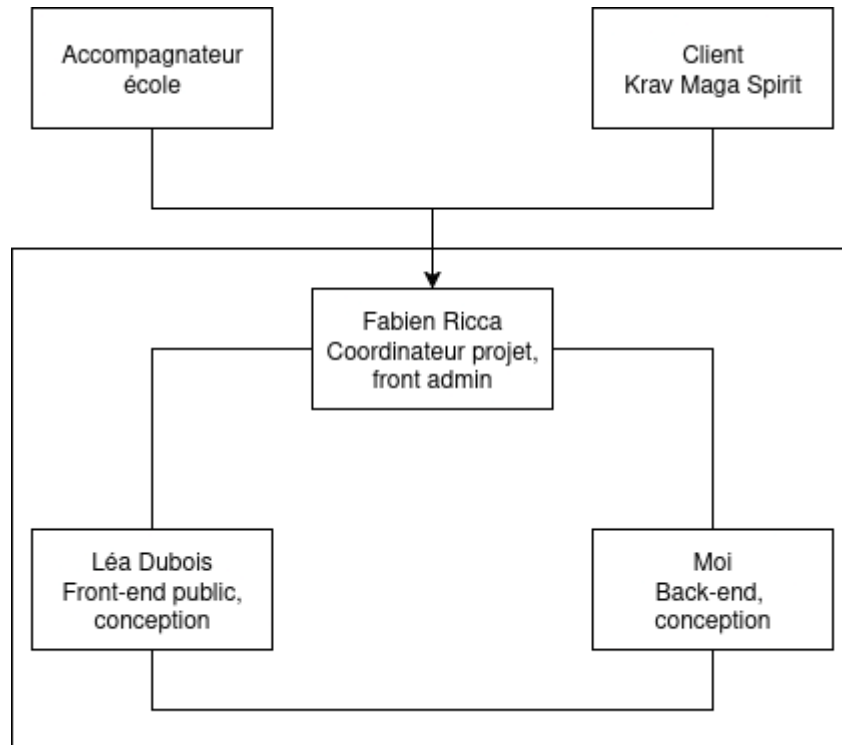
- **Réunion de lancement (lundi matin, semaine d'école)**
Objectif : planification des tâches, répartition du travail.
- **Réunion de clôture (vendredi après-midi, semaine d'école)**
Objectif : faire le point sur les réalisations, préparer le sprint suivant.
- **Communication asynchrone (quotidienne)**
Outil : Google chat
Objectif : coordination continue, résolution des blocages.
- **Points avec le club (à chaque phase clé)**
Objectif : présentation des maquettes, validation des fonctionnalités, retours utilisateurs.
- **Points pédagogiques (périodiques)**
Objectif : suivi du projet par les encadrants de formation, retour sur les choix techniques.

4. Acteurs responsables de la communication

- **Fabien R.** : responsable de la coordination générale, lien principal avec le client, animation des réunions.
- **Léa D.** : communication sur le front public, retour d'expérience utilisateur.
- **Corentin R.** : communication sur l'avancée du back-end, veille technique.

L'équipe projet

Organigramme



L'organigramme ci-dessous illustre la structure organisationnelle mise en place pour le projet de refonte du site web du club **Krav Maga Spirit**. Il met en évidence la hiérarchie fonctionnelle et les rôles de chacun des acteurs impliqués.

Parties prenantes externes

- **Client – Club Krav Maga Spirit** : Commanditaire du projet, il exprime les besoins, valide les livrables clés et s'assure de la cohérence globale par rapport aux attentes du club.
- **Accompagnateur École** : Référent pédagogique chargé du suivi académique du projet. Il encadre le travail des étudiants et valide la progression dans le cadre de la formation.

Équipe opérationnelle

- **Fabien R.** :
 - Rôle principal : gestion de projet, développement du front-end de l'espace administrateur, intégration des différentes parties.

- Assure le lien entre les parties prenantes et coordonne le bon déroulement du projet.
- **Léa D. :**
 - Rôle : développement de l'interface publique du site et participation à la conception globale.
 - Se concentre sur l'aspect visuel et l'expérience utilisateur.
- **Corentin R. :**
 - Rôle : développement back-end du site, participation à l'architecture technique.
 - Garant de la logique serveur, de la sécurité et des performances.

Ressources du projet

Le projet de refonte du site web du club **Krav Maga Spirit** mobilise un ensemble de ressources adaptées aux exigences techniques, humaines et organisationnelles d'un développement web moderne. Ces ressources sont structurées autour de plusieurs pôles :

Ressources matérielles

- **Postes de travail** : Chaque membre de l'équipe utilise son propre ordinateur personnel, équipé des outils nécessaires au développement. Cela assure une bonne maîtrise de l'environnement de travail et un accès constant aux ressources techniques.
- **Infrastructure de production** : Un serveur VPS hébergé chez OVH a été déployé pour l'hébergement de la version finale du site. Ce serveur, financé par Fabien Ricca, offre une autonomie totale et la souplesse nécessaire pour le déploiement continu.
- **Stockage en ligne** : Amazon S3 est utilisé pour héberger les fichiers et médias du site, garantissant performance, disponibilité et évolutivité du stockage.

Ressources humaines

- **Équipe projet** : Composée de trois développeurs, l'équipe travaille activement sur le projet lors des semaines de formation (1 semaine sur 5 en présentiel) et consacre également du temps personnel en dehors de l'école. Les rôles sont complémentaires, assurant une bonne répartition des compétences (coordination, front-end, back-end).

- **Encadrement** : L'équipe bénéficie d'un suivi régulier de l'encadrant pédagogique, qui accompagne la progression du projet et valide les étapes clés.

Ressources documentaires

- **Documents projet** : Un corpus documentaire a été élaboré, comprenant le cahier des charges, les spécifications fonctionnelles et techniques, ainsi que divers diagrammes UML pour structurer la conception du site.
- **Partage collaboratif** : Tous les documents sont centralisés sur Google Drive, garantissant l'accessibilité et la synchronisation entre les membres de l'équipe.

Ressources techniques

- **Outils de développement** : L'équipe utilise des outils reconnus dans l'industrie, tels que des IDE adaptés aux technologies employées, DBeaver pour la gestion des bases de données, et Git/GitLab pour le versionnement du code.
- **Environnements de déploiement** :
 - En local pour le développement et les tests.
 - Sur serveur VPS distant pour la production, avec une chaîne CI/CD mise en place.
 - Une séparation stricte est respectée entre les environnements pour assurer la stabilité et la sécurité.
- **Logiciels et licences** : Le projet repose sur des solutions open source ou gratuites, afin de respecter les contraintes budgétaires sans compromettre la qualité.

Ressources financières

- Le projet, mené bénévolement, est autofinancé à minima :
 - Hébergement VPS et nom de domaine pris en charge personnellement.
 - Utilisation d'Amazon S3 avec des coûts variables selon la consommation réelle.
 - Acquisition de certificats SSL pour sécuriser l'accès au site.

RACI du projet

Le RACI est un outil de gestion de projet utilisé pour définir clairement les rôles et responsabilités des différentes parties prenantes sur chaque activité. L'acronyme RACI signifie :

- **R – Responsable (Responsable)** : la personne qui exécute la tâche. Il peut y avoir plusieurs responsables.
- **A – Accountable (Autorité / Responsable final)** : la personne qui prend la décision finale et valide le résultat. Il ne peut y avoir qu'un seul "A" par activité.
- **C – Consulted (Consulté)** : les personnes dont l'avis est sollicité avant ou pendant la réalisation de la tâche.
- **I – Informed (Informé)** : les personnes à tenir informées de l'avancement ou du résultat de l'action.

Activités	Fabien R.	Moi	Léa D.	Accompagnateur	KMS (client)
Collecte des besoins	R	C	C	A	C
Spécification	R	C	C	A	A
Conception	R	C	C	A	C
Réalisation Front Admin	R	I	C	A	I
Réalisation front public	C	I	R	A	I
Réalisation back end	C	R	I	A	I
Integration	R	C	C	A	I
Tests	R	C	C	A	I
Déploiement	R	C	C	A	I
Validation client	C	I	I	I	A
Formation utilisateurs	R/A	C	C	C	A
Suivi et évaluation	R	C	C	C	I
Communication avec les parties prenantes	R	C	C	A	I
Documentation	R	C	C	A	I

Cet outil permet d'assurer une répartition claire des responsabilités, de fluidifier la communication et de limiter les confusions dans la gestion de projet.

Analyse des responsabilités par profil

Fabien R.(Coordinateur projet / Front-end Admin / Intégration)

Il est responsable de la majorité des activités de développement, notamment l'intégration, la gestion du front-end administrateur, les tests et le déploiement. Il coordonne également l'ensemble du projet et assure le lien avec les parties prenantes. Son rôle central se positionne à la fois comme acteur technique et référent organisationnel.

Léa D. (Front-end Public / Conception)

Elle est responsable de la conception du front-end public. Elle participe activement à la spécification, aux tests, à la documentation et à la formation des utilisateurs. Son rôle est essentiel pour assurer une interface accessible et fonctionnelle destinée aux visiteurs du site.

Corentin R. (Back-end / Conception)

Je suis chargé du développement du back-end et intervient également dans le développement de la partie admin. Je contribue à l'intégration, au déploiement et à la structuration technique du projet. Mon expertise est essentielle à la robustesse et à la sécurité de l'application.

Accompagnateur (Accompagnateur école)

Il agit en tant que référent académique. Il est consulté à toutes les étapes clés et joue un rôle d'autorité dans la validation de certaines phases (spécification, formation). Il guide le projet en veillant à sa cohérence avec les attentes pédagogiques.

KMS (Client - Club Krav Maga Spirit)

Le client est principalement consulté lors de la collecte des besoins, des spécifications et des validations. Il est informé tout au long du processus de développement. Il valide les livrables finaux. Sa participation garantit que la solution réponde fidèlement à ses besoins.

Analyse des risques

Evaluation des risque

Risques	Fréquence	Gravité	Détection	Criticité
Incompatibilité techniques entre les technologies choisies et l'environnement existant	3	4	3	36
Retard dans l'exécution des tâches qui pourraient affecter le calendrier du projet	3	4	3	36
Indisponibilité des API interrogés	3	4	2	24
Manque d'engagement ou de disponibilité d'une équipe impliqué	2	4	2	16
Communication insuffisante ou inefficace entre les différentes parties prenantes	2	3	2	12
Perte de données ou corruption du code source	2	5	1	10
Défaillance de l'infrastructure d'hébergement (VPS OVH)	2	3	1	6
Incompatibilité entre les environnements de développement des membres de l'équipe	2	2	1	4
Catastrophe naturelle (inondation, incendie)	1	5	1	5

Évaluation des risques – Explication des colonnes

L'évaluation des risques permet d'anticiper les problèmes potentiels pouvant affecter le bon déroulement du projet. Chaque risque est évalué selon quatre critères :

Fréquence : Probabilité d'apparition du risque (de 1 à 5)

1 = Très rare, 5 = Très fréquent

Gravité : Impact du risque sur le projet s'il se réalise (de 1 à 5)

1 = Impact faible, 5 = Impact critique

Détection : Capacité à détecter le risque avant qu'il n'ait un effet (de 1 à 5)

1 = Très facilement détectable, 5 = Difficile à détecter

Criticité : Niveau de priorité du risque à traiter. Elle est calculée par la formule :

Fréquence × Gravité × Détection

Plus le score est élevé, plus le risque est critique.

Risques à criticité élevée

Les risques à criticité élevée sont ceux dont le niveau de menace, évalué selon les critères de fréquence, gravité et détectabilité, dépasse un seuil critique. Dans le cas du projet de refonte du site KMS, deux risques majeurs ont été identifiés comme prioritaires en raison de leur criticité élevée (note de 36) : **l'incompatibilité technique entre les technologies choisies et l'environnement existant** ainsi que **les retards dans l'exécution des tâches impactant le calendrier global**.

Ces risques combinent une fréquence élevée, une gravité importante et une difficulté de détection, ce qui augmente leur potentiel de perturbation sur le projet. Une mauvaise intégration technique pourrait remettre en question certains choix d'architecture, nécessitant des ajustements de dernière minute. Quant aux retards, ils risquent d'entraîner un effet de cascade sur l'ensemble du planning, en compromettant les délais de livraison déjà stricts. Ces risques nécessitent donc une surveillance rapprochée et des plans d'action anticipés, incluant une planification réaliste, une validation régulière des choix techniques et un suivi hebdomadaire de l'avancement.

Risques à criticité modérée

Les risques classés à criticité modérée (notes comprises entre 10 et 29) représentent une menace significative, bien que moins urgente que les précédents. Ils incluent notamment **l'indisponibilité des API tierces** (24), **le manque d'engagement ou de disponibilité d'un membre de l'équipe** (16), **la communication inefficace entre les parties prenantes** (12), et **la perte de données ou la corruption du code source** (10).

Ces risques peuvent être relativement fréquents et parfois complexes à détecter, sans pour autant entraîner des conséquences immédiates aussi critiques. Ils nécessitent une organisation rigoureuse, standardisation des outils de communication et clarification des rôles au sein de l'équipe, ou encore l'adoption de bonnes pratiques de sauvegarde et d'utilisation d'un gestionnaire de versions comme Git. S'ils sont bien anticipés, ces risques peuvent être efficacement contrôlés et limités dans leurs impacts.

Risques à criticité faible

Enfin, les risques à criticité faible (score inférieur à 10) sont considérés comme ayant un impact limité et une faible probabilité d'occurrence. Dans ce projet, cela inclut **la défaillance de l'infrastructure d'hébergement** (6), **l'incompatibilité entre les environnements de développement des différents membres de l'équipe** (4), et **la survenue d'une catastrophe naturelle** (5).

Bien que ces risques ne nécessitent pas une vigilance constante, il est néanmoins utile de les prendre en compte dans une logique de prévention. Par exemple, choisir un hébergeur fiable comme OVH, documenter précisément l'environnement technique à reproduire, ou encore conserver des sauvegardes hors ligne permettent

de minimiser les effets si ces scénarios venaient à se produire. Une stratégie de gestion des incidents, même basique, renforce ainsi la résilience du projet sans nécessiter de lourds investissements.

Plan d'action

Risques	Actions
Incompatibilité techniques entre les technologies choisies et l'environnement existant	Un risque majeur pouvant bloquer le déploiement. Il est anticipé par des tests réguliers sur le VPS, une documentation rigoureuse des choix techniques et des alternatives prêtes à l'emploi si une technologie pose problème.
Retard dans l'exécution des tâches qui pourraient affecter le calendrier du projet	planification précise, un suivi régulier via un outil collaboratif et une répartition équilibrée des tâches ont été mis en place. Utilisation des méthodes agile. En cas de blocage, des ajustements sont prévus.
Indisponibilité des API interrogés	L'arrêt temporaire d'une API peut perturber le fonctionnement. Pour limiter l'impact, des caches, des alternatives (mock ou API de secours) .
Manque d'engagement ou de disponibilité d'une équipe impliqué	Une équipe réduite rend ce risque sensible. Il est traité par une communication claire, la polyvalence des membres et une documentation continue pour faciliter la reprise.
Communication insuffisante ou inefficace entre les différentes parties prenantes	Des malentendus avec le client ou dans l'équipe peuvent créer des erreurs. Des points réguliers, des comptes rendus et des outils partagés permettent de maintenir une bonne synchronisation.
Perte de données ou corruption du code source	Le code est protégé par un usage rigoureux de Git, des sauvegardes fréquentes et une organisation stricte des branches de développement.
Défaillance de l'infrastructure d'hébergement (VPS OVH)	Un problème d'hébergement peut interrompre le service. Ce risque est limité par le choix d'un hébergeur fiable et la mise en place de sauvegardes.
Incompatibilité entre les environnements de développement des membres de l'équipe	Des erreurs liées à des différences de configuration sont évitées grâce à une standardisation des outils grâce à Docker et une documentation technique commune.
Catastrophe naturelle (inondation, incendie)	Peu probable mais impactant. Des copies de sécurité, l'hébergement cloud et le travail à distance assurent une reprise rapide en cas de crise.

Le **coordinateur du projet** est en charge du suivi global du plan d'action lié aux risques. Une revue hebdomadaire est réalisée chaque lundi lors des réunions d'équipe afin d'évaluer l'évolution des risques identifiés et d'en détecter de nouveaux. Chaque membre de l'équipe a la responsabilité de signaler immédiatement toute alerte ou indicateur de risque en lien avec ses missions.

En cas de survenue d'un problème, les actions correctives suivent une **procédure d'escalade progressive** : tentative de résolution au sein de l'équipe, intervention du

coordinateur si nécessaire, puis recours à l'encadrement académique. Le client n'est informé que si l'impact touche les livrables ou le planning de livraison.

Ce plan est un document **vivant**, régulièrement ajusté au fil de l'avancement du projet et enrichi en cas d'identification de nouveaux risques.

Efficiency

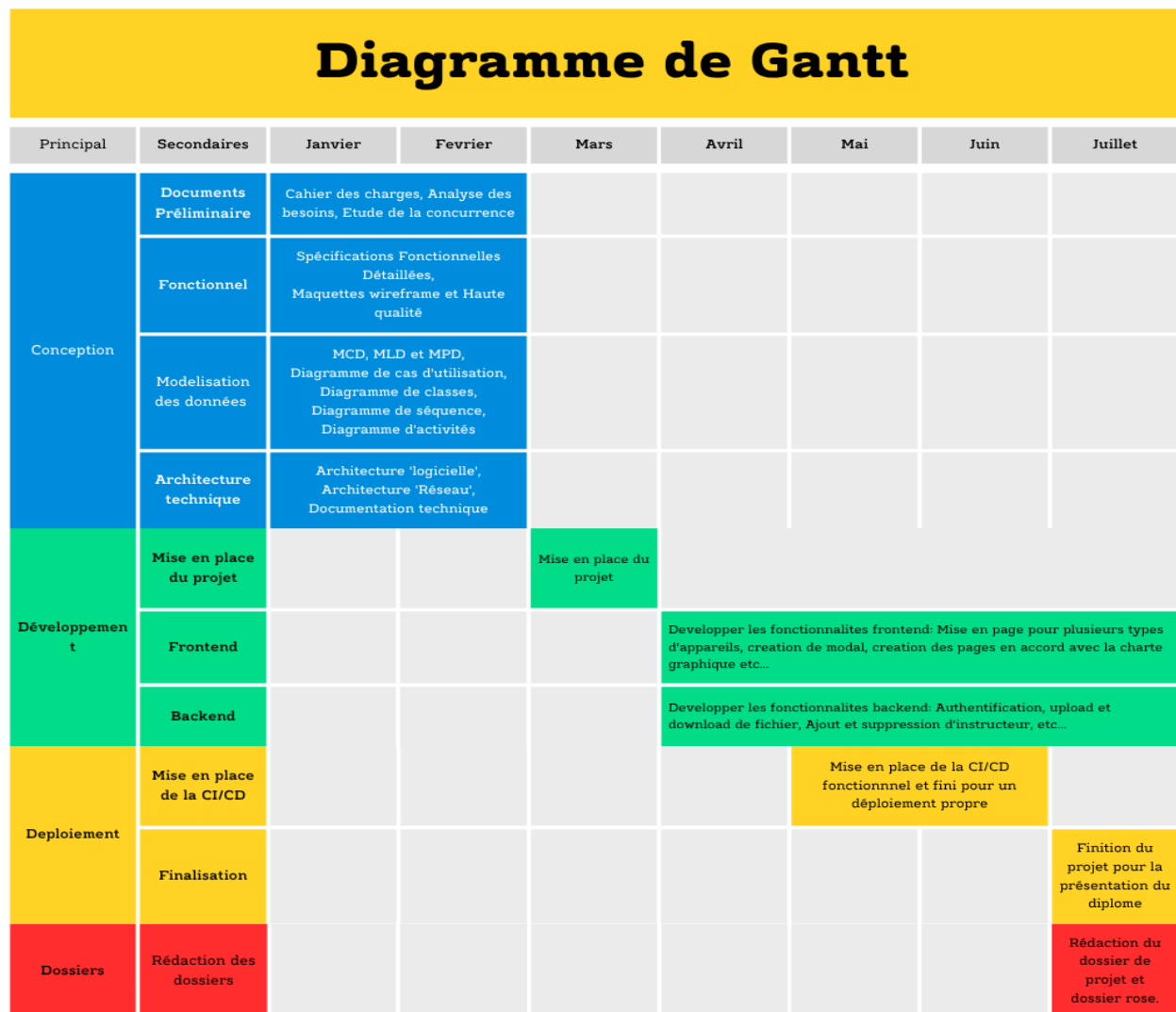
Le tableau d'efficacité a pour but d'évaluer la capacité du projet à atteindre ses objectifs en optimisant l'utilisation des ressources disponibles. Il permet d'analyser de manière synthétique plusieurs critères essentiels, comme la gestion du temps, des coûts, la répartition des compétences, la qualité du livrable ou encore la performance technique. Cette analyse fournit une vision globale de l'efficacité du projet, en mettant en lumière les points forts et les aspects nécessitant une vigilance ou une amélioration continue.

Critère évalué	indicateur d'efficacité	Evaluation	Justification
Utilisation des ressources humaines	Répartition équilibrée des tâches selon les compétences	Efficace	L'équipe est réduite mais polyvalente, chaque membre gère une partie spécifique.
Temps de développement	Respect du planning prévisionnel	Moyennement efficace	Le projet a connu un léger décalage, mais l'échéance finale est toujours tenable.
Coût du projet	Dépenses limitées / respect du budget	Très efficace	Aucun coût salarié, infrastructure financée à titre personnel, usage d'outils gratuits.
Qualité du livrable	Conformité aux exigences / peu de retours correctifs	En cours d'évaluation	Les retours intermédiaires sont positifs ; évaluation finale prévue à la livraison.
Réutilisabilité / évolutivité de la solution	Capacité à intégrer de futures fonctionnalités	Efficace	L'architecture choisie permet d'ajouter de nouvelles fonctionnalités facilement.
Performance technique	Temps de réponse, disponibilité, stabilité	Efficace	Les tests montrent de bonnes performances, notamment grâce au choix d'OVH et S3.
Communication et coordination	Fréquence des échanges et suivi de projet	Moyennement efficace	L'alternance limite les réunions, mais un effort d'organisation est constant.

Dans l'ensemble, le projet a été mené avec une efficacité satisfaisante malgré certaines contraintes, notamment liées au temps limité imposé par le rythme d'alternance. La répartition des rôles au sein de l'équipe a permis une bonne complémentarité des compétences et un avancement structuré des tâches. Les

choix techniques, orientés vers des solutions open source, ont permis de respecter un budget quasi nul tout en assurant une performance correcte. Quelques problèmes ont retardé la livraison finale, une date revue à la hausse a été acceptée. Cette rétrospective met en évidence l'importance de l'organisation, de l'adaptabilité de l'équipe, et de la clarté des objectifs dès le début du projet.

Planification



Conception

1. Documents Préliminaires (Janvier)

- **Cahier des charges** : Ce document définit les besoins et les attentes du client. Il décrit les fonctionnalités attendues, les contraintes techniques et les objectifs du projet.
 - **Analyse des besoins** : Cette étape consiste à recueillir et analyser les besoins des utilisateurs finaux pour s'assurer que le produit final répondra à leurs attentes.
 - **Étude de la concurrence** : Analyser les produits concurrents pour identifier leurs forces et faiblesses, et déterminer comment votre produit peut se différencier.
2. Fonctionnel (Janvier)
- **Spécifications Fonctionnelles** : Décrire en détail les fonctionnalités du produit, les interactions utilisateur, et les workflows.
 - **Détail des maquettes wireframe et haute qualité** : Créer des maquettes pour visualiser l'interface utilisateur et obtenir des retours précoces.
3. Modélisation des données (Janvier)
- **MCD, MLD et MPD** : Créer des modèles conceptuels, logiques et physiques de données pour structurer la base de données.
 - **Diagrammes de cas d'utilisation, de classes, de séquence, et d'activités** : Ces diagrammes aident à visualiser les interactions entre les différents composants du système et les processus métiers.
4. Architecture technique (Janvier)
- **Architecture logique, Architecture Réseau, Documentation technique** : Définir l'architecture globale du système, y compris les choix technologiques, les schémas réseau et la documentation technique associée.

Développement

1. Mise en place du projet (Février)
 - Configurer l'environnement de développement, les outils de gestion de projet, et les dépôts de code source.
2. Frontend (Mars à Mai)
 - Développer les fonctionnalités frontend : Mettre en page pour plusieurs types d'appareils, créer des modals, et des pages en accord avec la charte graphique, etc.
3. Backend (Mars à Mai)
 - Développer les fonctionnalités backend : Gérer l'authentification, l'upload et le download de fichiers, ajouter et supprimer des instructeurs, etc.

Déploiement

1. Mise en place de la CI/CD (Mai)
 - Configurer l'intégration continue et le déploiement continu pour automatiser les tests et les déploiements.
2. Mise en place du déploiement (Juin)
 - Préparer et exécuter le déploiement de l'application en environnement de production.

Finalisation

1. Finalisation (Juin à Juillet)
 - **Finition du projet** : Effectuer les derniers ajustements et corrections.
 - **Préparation du diplôme** : Préparer les documents nécessaires pour la présentation du diplôme.

Dossiers

1. Rédaction des dossiers (Juillet)
 - **Rédaction du dossier de projet et du dossier rose** : Rédiger les documents finaux détaillant le projet et les aspects administratifs.

Conception et réalisation :

Présentation de l'existant

Le site web actuel du **club Krav Maga Spirit**, accessible à l'adresse www.krav-maga-spirit.fr, a été développé il y a plusieurs années. Il reflète aujourd'hui un écart important avec les standards actuels du web, tant sur le plan technique que fonctionnel. Cette situation limite fortement la communication du club, sa visibilité en ligne, ainsi que la capacité de ses responsables à gérer efficacement le contenu du site.

Limites techniques

Le site repose sur une architecture technique devenue obsolète. Il n'intègre pas de framework moderne, ce qui rend le code difficile à maintenir, peu évolutif, et peu performant, notamment sur les terminaux mobiles. Son design est responsive mais il rassemble trop d'informations ce qui le rend illisible : il s'adapte mal aux différentes tailles d'écran, ce qui nuit à l'expérience utilisateur sur smartphones et tablettes. On constate également des temps de chargement élevés et un manque d'optimisation des ressources (images, scripts), ce qui dégrade à la fois la fluidité de navigation et le référencement naturel du site.

Faiblesses fonctionnelles

Plusieurs fonctionnalités essentielles sont limitées ou absentes. L'interface d'administration est peu ergonomique, rendant difficile la mise à jour régulière du contenu. La gestion des albums photos, documents administratifs ou présentations de l'équipe nécessite souvent des manipulations techniques, peu accessibles pour des utilisateurs non spécialisés. De plus, la navigation manque de clarté et ne met pas efficacement en valeur les informations prioritaires comme les horaires, les événements ou les contacts.

Impact sur la communication

Ces limites techniques et fonctionnelles ont un impact direct sur l'image du club et sur la qualité de sa communication numérique. Le site ne reflète pas l'identité dynamique du club Krav Maga Spirit. La difficulté à maintenir le contenu à jour conduit à une obsolescence des informations publiées. L'expérience utilisateur est pénalisée par la lenteur, la structure confuse et le manque d'adaptabilité mobile, ce qui peut dissuader de potentiels adhérents de prendre contact ou de s'inscrire.

Besoins non satisfaits

Les échanges avec l'équipe dirigeante ont mis en évidence plusieurs besoins urgents non couverts par la solution actuelle :

- une interface d'administration simple, permettant une autonomie totale des gestionnaires du site ;
- une présentation flexible des contenus, afin d'adapter facilement les informations en fonction de l'actualité du club ;
- une optimisation mobile indispensable pour répondre aux usages actuels ;
- une plateforme performante, stable et moderne, capable d'évoluer dans le temps.

Opportunités d'amélioration

Face à ce constat, le projet de refonte du site vise une transformation complète, avec les priorités suivantes :

- l'adoption de technologies modernes (Angular, Node.js, etc.) pour assurer performance et maintenabilité ;
- la mise en place d'une nouvelle interface d'administration intuitive et accessible, adaptée aux non-techniciens ;
- la refonte complète de l'expérience utilisateur, avec une navigation claire, un design moderne et une parfaite compatibilité mobile ;
- l'amélioration de la communication digitale du club, avec des outils modernes de présentation, de contact et de valorisation de ses activités.

Ce projet représente donc une opportunité de redéfinir l'image numérique du club Krav Maga Spirit en la rendant plus cohérente, plus efficace et plus attractive.

Choix des technologie et justification

Pour répondre aux enjeux de performance, de maintenabilité et de scalabilité du nouveau site du club Krav Maga Spirit, nous avons sélectionné une stack technologique moderne, cohérente et éprouvée. Le choix des outils repose à la fois sur leur compatibilité avec les besoins fonctionnels du projet, leur adoption large dans l'écosystème du développement web, et leur facilité de prise en main pour l'équipe de développement.

- **Frontend :**

Le framework Angular a été retenu pour la réalisation de l'interface utilisateur. Il permet de créer des interfaces dynamiques, modulaires et responsives. Associé à TypeScript, il offre une structure rigoureuse du code, améliore la maintenabilité et facilite la détection des erreurs lors du développement.

- **Backend :**

Le serveur est développé en Node.js avec le framework Express.js, qui fournit une architecture légère, rapide et extensible pour gérer les routes, les requêtes API, l'authentification, et la logique métier du site. Cette combinaison assure un bon équilibre entre performance, rapidité de développement et flexibilité, comme le front et le back s'appuie sur le même langage socle (JavaScript) cela simplifie une mise en œuvre de type Full Stack.

- **Microservice d'authentification :**

Un microservice indépendant dédié à la gestion des comptes et de l'authentification a été conçu en Go, reconnu pour sa rapidité d'exécution et sa légèreté. Il permet de sécuriser les accès et d'assurer une bonne séparation des responsabilités.

- **Bases de données :**

Deux types de bases ont été retenus selon les besoins :

- Une base relationnelle PostgreSQL pour stocker les données structurées liées aux utilisateurs, aux horaires de cours, aux inscriptions, etc.
- Une base NoSQL MongoDB utilisée par le microservice Go, adaptée aux formats de données plus flexibles.

- **Déploiement et conteneurisation :**

L'ensemble des services est conteneurisé avec Docker, afin de garantir une portabilité, une isolation des environnements et une facilité de déploiement identique entre développement et production.

- **CI/CD (Intégration et déploiement continu) :**

Nous utilisons GitLab CI/CD pour automatiser le build et le déploiement à chaque commit. Cela permet de sécuriser la qualité du code et d'accélérer le cycle de livraison tout en réduisant les risques d'erreur humaine.

- **Stockage des fichiers :**

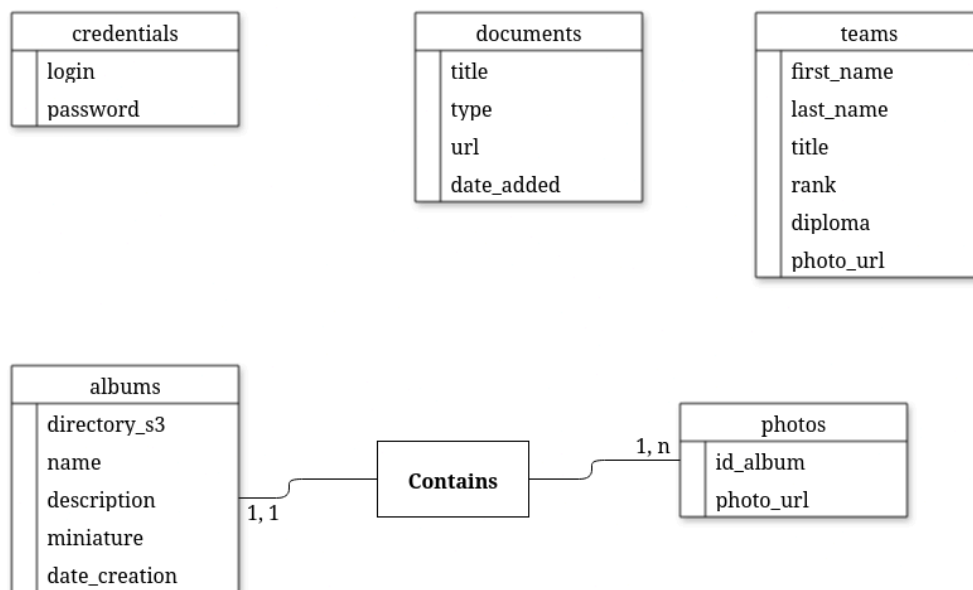
Pour héberger les fichiers statiques (documents PDF, images, photos

d'événements...), nous utilisons Amazon S3, un service de stockage cloud scalable, sécurisé et hautement disponible.

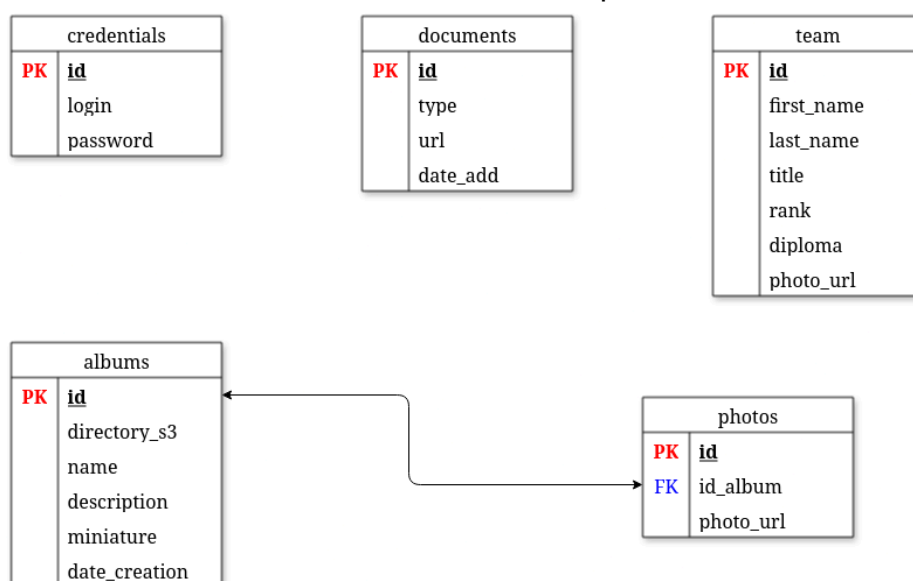
Conception de la base de données

La conception de la base de données semble suivre une approche structurée pour gérer les informations relatives aux utilisateurs, aux documents, aux équipements et aux albums photo. Voici un aperçu des principaux éléments :

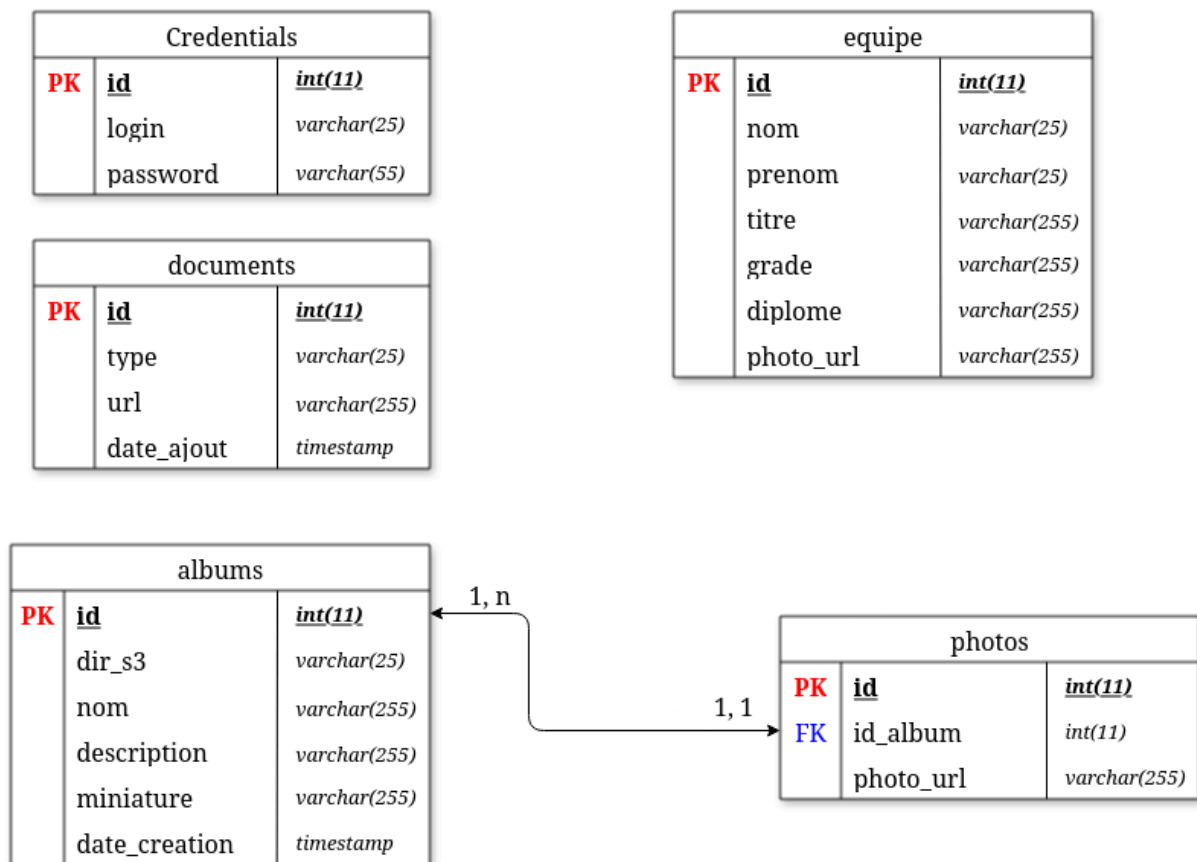
1. **MCD** (Cette image représente le Modèle Conceptuel de Données (MCD). Elle montre les entités principales du système.) :



2. **MLD** (Cette image illustre le Modèle Logique de Données (MLD). Elle détaille les entités et leurs attributs, ainsi que les relations entre elles.) :



3. **MPD** (Cette image montre le Modèle Physique de Données (MPD). Elle inclut des détails spécifiques sur les types de données et les tailles des champs pour chaque attribut dans les tables.) :



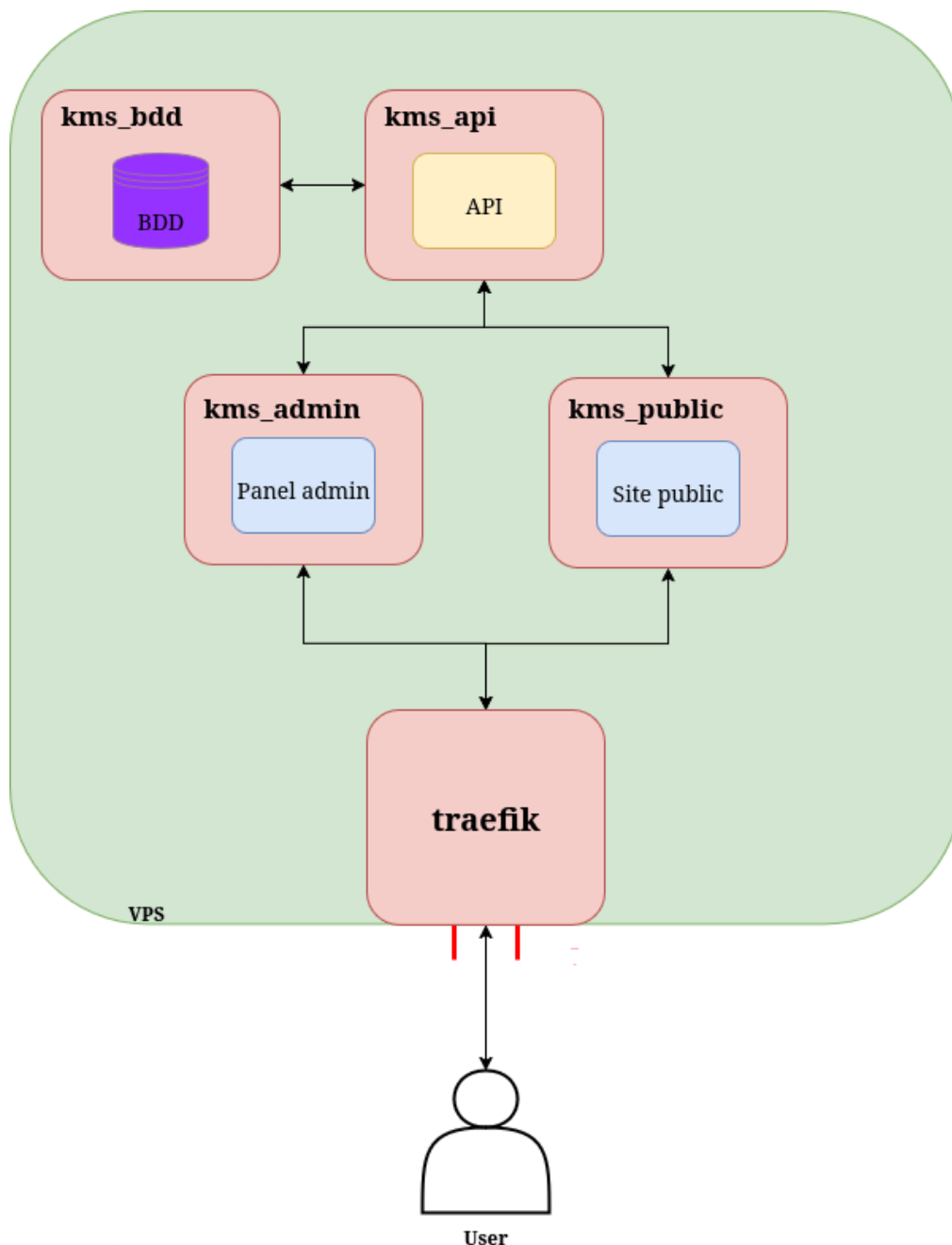
4. NoSQL

Credentials		
PK	<u>UniqueID</u>	<u>ObjectID</u>
	username	String
	password	String
	jwt_refresh_token	String
	create_at	Date
	updated_at	Date

Architecture de l'application

L'application du projet KMS repose sur une architecture logicielle moderne, modulaire et évolutive, conçue selon les principes des microservices et déployée sous forme de conteneurs. Cette approche vise à garantir une séparation claire des responsabilités, à simplifier la maintenance et à faciliter les mises à jour, tout en assurant des performances fiables et une sécurité adaptée aux besoins du club Krav Maga Spirit.

L'architecture suit un modèle client-serveur distribué, déployé sur un serveur VPS, tout en s'appuyant sur des services cloud (notamment Amazon S3) pour optimiser les ressources, réduire les coûts et bénéficier d'une scalabilité à la demande. Ce choix hybride permet de concilier autonomie d'infrastructure et efficacité des services managés.



Architecture logique

Le système est structuré en **quatre couches logiques** qui interagissent de manière fluide :

- **Couche de présentation** : Deux interfaces Angular distinctes sont prévues, chacune adaptée à un public cible :
 - **kms_public** : une interface dédiée au grand public et aux adhérents, mettant en avant les contenus essentiels (galeries, équipe, documents, etc.) via une SPA (Single Page Application) fluide et responsive.
 - **kms_admin** : une interface d'administration réservée aux responsables du club, offrant des fonctionnalités complètes de gestion de contenu. Cette séparation assure une sécurité renforcée et une meilleure ergonomie pour chaque profil utilisateur.
- **Couche API** : Une API REST unique, développée en Node.js avec Express.js, centralise toutes les opérations métier. Elle assure la communication entre les interfaces front-end et la couche de données, tout en gérant l'authentification et les règles de gestion.
- **Couche de données** :
 - Les données structurées sont stockées dans une base de données relationnelle PostgreSQL et une base de données non relationnelles MongoDB, garantissant robustesse et intégrité.
 - Les fichiers multimédias (photos, documents PDF, etc.) sont gérés via Amazon S3, pour un stockage scalable, sécurisé et performant.
- **Couche d'infrastructure** : Un reverse proxy Traefik est utilisé pour le routage des requêtes, la gestion des certificats SSL (HTTPS) et l'équilibrage de charge entre les différents services, assurant ainsi une distribution optimale du trafic.

Composants techniques principaux

- **Front-end Angular** : L'architecture component-based d'Angular permet une modularité forte. Chaque interface (publique et admin) est conçue comme une application SPA, garantissant une navigation fluide et une réactivité optimale.

- **Back-end Node.js/Express.js** : Le serveur d'application traite toutes les requêtes API, assure la transformation des données, et utilise un système JWT (JSON Web Token) pour sécuriser les accès. Il se charge également de la communication avec PostgreSQL et l'intégration avec Amazon S3 pour le traitement des fichiers (upload, compression, distribution).
- **Conteneurisation avec Docker** : Tous les services sont isolés et déployés sous forme de conteneurs Docker, facilitant les déploiements, les tests et la montée en charge.
- **CI/CD avec GitLab** : L'intégration continue est assurée via **GitLab CI/CD**, permettant de tester, valider et déployer automatiquement les évolutions à chaque mise à jour du code, garantissant ainsi une livraison rapide et fiable.

Sécurité de l'application

La sécurité de l'application KMS a été pensée comme un élément fondamental dès la phase de conception. Une stratégie de défense en profondeur a été adoptée, intégrant des mécanismes de protection à tous les niveaux : du code source jusqu'à l'infrastructure réseau, en passant par la gestion des accès, des données et des communications.

Authentification sécurisée avec JWT

L'accès à l'interface d'administration est protégé par un système d'authentification basé sur les JSON Web Tokens (JWT) :

- **Génération sécurisée** : un token signé est émis après authentification, contenant uniquement les informations essentielles (ID utilisateur, rôle, expiration).
- **Expiration et renouvellement** : les tokens ont une durée de vie limitée à 2 heures.
- **Validation systématique** : chaque requête vers l'API vérifie la validité, la signature et la date d'expiration du token.

Gestion des mots de passe

Les mots de passe utilisateurs sont gérés selon les meilleures pratiques :

- **Hashage avec bcrypt** : aucun mot de passe n'est stocké en clair. Le hachage est renforcé par un facteur de coût élevé.
- **Politique de sécurité** : complexité obligatoire (longueur, caractères spéciaux), avec contrôle en amont lors de la création des comptes

administrateurs.

Chiffrement des communications via Traefik

L'ensemble des échanges entre le client et le serveur est protégé par HTTPS :

- **Certificats SSL automatiques** : délivrés et renouvelés par Let's Encrypt via Traefik.
- **Terminaison SSL** : le reverse proxy déchiffre les connexions sécurisées et les achemine vers les services internes via un réseau Docker isolé.
- **Redirection et PFS** : redirection automatique de HTTP vers HTTPS, avec prise en charge du Perfect Forward Secrecy pour une sécurité renforcée.

Sécurisation du réseau et des conteneurs Docker

- **Isolation réseau** : les services communiquent uniquement sur un réseau interne Docker privé, inaccessible depuis l'extérieur.
- **Exposition minimale** : seul Traefik est exposé à Internet, limitant ainsi la surface d'attaque.

Protection des accès à la base de données avec Prisma ORM

L'utilisation de Prisma permet de sécuriser toutes les interactions avec la base de données PostgreSQL :

- **Requêtes préparées automatiquement** : neutralisation des injections SQL.
- **Vérification des types et validation des schémas** : réduction des erreurs et des entrées malveillantes.
- **Gestion des connexions sécurisée** : avec limitations, timeouts et logs d'audit pour garantir traçabilité et stabilité.

Protection des informations sensibles

- **Variables d'environnement** : toutes les données sensibles (clés, mots de passe, tokens) sont stockées de manière sécurisée, hors du code source.
- **Rotation régulière** des secrets pour limiter les risques en cas de compromission.
- **Isolation stricte des environnements** : développement, test et production sont totalement séparés.

Sécurité des fichiers et d'Amazon S3

- **Politiques IAM minimales** : les permissions AWS sont limitées au strict nécessaire.
- **Validation des fichiers uploadés** : contrôle du type MIME, taille et intégrité.
- **Utilisation de clés dédiées** sécurisées et renouvelées régulièrement.

Prévention des injections et attaques XSS

- **Protection native via Prisma** contre les injections SQL.
- **Validation et sanitisation des données** côté serveur et côté client.
- **Politique CSP** mise en place pour bloquer l'exécution de scripts non autorisés.
- **Angular** assure automatiquement une forte protection XSS grâce à sa stratégie de binding sécurisé.

Mise à jour et maintenance régulière

- **Surveillance des vulnérabilités** via npm audit et GitLab CI/CD.
- **Dépendances maintenues à jour**, y compris les images Docker (possibles utilisation de snyk pour prévenir les failles des images docker).
- **Veille de sécurité** sur l'ensemble des composants utilisés.

Sauvegarde et plan de reprise

- **Sauvegardes régulières** des bases de données et des fichiers.
- **Plan de reprise d'activité (PRA)** défini et testé, permettant de restaurer rapidement le service en cas d'incident majeur.

Présentation des outils

Dans le cadre du développement de l'application KMS, plusieurs outils ont été mobilisés afin de répondre aux besoins techniques, collaboratifs et organisationnels du projet.

J'ai utilisé **NeoVim** comme éditeur de code principal tout au long du projet. Sa légèreté, sa rapidité d'exécution et sa haute personnalisation m'ont permis de créer un environnement de développement adapté à mes besoins. Grâce à l'utilisation de plugins comme *nvim-treesitter*, *LSP* (Language Server Protocol), et *telescope*, j'ai pu bénéficier de fonctionnalités avancées comme l'autocomplétion intelligente, la navigation rapide dans le code, la coloration syntaxique moderne et le linting en temps réel. Cette configuration m'a offert un confort de développement équivalent aux IDE classiques tout en conservant la performance et la simplicité d'un éditeur en ligne de commande.

J'ai utilisé GitLab comme plateforme complète de gestion de projet. Au-delà de la gestion du code source avec Git, j'ai tiré parti de GitLab CI/CD pour automatiser le cycle de vie des déploiements. Les pipelines définis dans les fichiers `.gitlab-ci.yml` m'ont permis de déclencher automatiquement les étapes de test, de build, et de mise en production dès qu'une nouvelle modification était poussée sur la branche principale. GitLab m'a également permis d'organiser les tâches à réaliser avec le tableau *issues* et *milestones*, assurant ainsi une bonne planification, répartition et traçabilité du travail d'équipe.

Docker a été utilisé pour containeriser l'ensemble des composants de l'application : API, bases de données, services front-end, Traefik et microservice d'authentification. Chaque composant est défini dans un container isolé, ce qui facilite les déploiements, les tests et la reproductibilité de l'environnement. Grâce à *Docker Compose*, j'ai pu orchestrer et configurer les services dans un seul fichier, rendant l'infrastructure simple à déployer sur n'importe quel VPS ou environnement cloud. Cette architecture containerisée a permis une meilleure gestion des dépendances, une sécurité accrue et une grande portabilité de l'application.

Prisma ORM a été utilisé pour gérer la couche d'abstraction entre le code Node.js et la base de données PostgreSQL. Prisma facilite la gestion des entités, des relations, et des migrations de schéma via un langage déclaratif clair. Il m'a permis de générer automatiquement des fonctions typesafe pour interroger la base, réduisant ainsi les risques d'erreurs au moment de l'exécution. Prisma intègre également un système de migration très pratique pour versionner l'évolution du schéma de données, ce qui a grandement facilité le travail d'équipe et les déploiements. En outre, Prisma contribue à la sécurité de l'application en générant des requêtes préparées, ce qui empêche les injections SQL.

Figma a été utilisé pour concevoir les maquettes graphiques des interfaces utilisateur. Cet outil de design collaboratif a facilité les échanges entre les membres de l'équipe autour de l'ergonomie et de l'expérience utilisateur.

Node.js, associé à son gestionnaire de paquets npm, a constitué la base technique du back-end. Cet environnement JavaScript orienté événement a permis de construire une API REST efficace, notamment grâce à Express.js.

Angular et Angular CLI ont été utilisés pour développer les interfaces front-end. Grâce à ce framework, j'ai pu créer deux applications SPA distinctes (publique et administrateur) avec un code modulaire et une navigation fluide.

Google Chat a été l'outil de communication principal de l'équipe. Il a permis d'échanger rapidement sur l'avancement du projet, de poser des questions techniques et d'assurer un suivi régulier entre les membres.

Google Drive a servi de plateforme de stockage pour centraliser tous les fichiers du projet : documentations, captures d'écran, exports, ressources de présentation, etc.

Google Docs a été utilisé pour rédiger les documents collaboratifs du projet, tels que les comptes-rendus, les descriptions techniques ou les plannings. Il a facilité le travail à plusieurs et la validation des livrables.

Draw.io a été mobilisé pour la création de schémas techniques, notamment l'architecture logicielle, les diagrammes UML ou les modèles de données, afin d'avoir une vision claire et structurée du système.

Enfin, Canva a permis de concevoir les tableaux utilisés sur ce même dossier de projet, notamment les slides destinées à l'oral, avec des visuels clairs et professionnels adaptés à une soutenance.

Conception

Diagramme de cas d'utilisation – Administrateur

Résumé :

Ce diagramme présente l'ensemble des interactions possibles entre l'administrateur et les différentes fonctionnalités de l'application KMS. Il offre une vue globale des responsabilités de l'utilisateur principal du back-office.



Description technique :

L'administrateur interagit avec quatres modules principaux :

Authentification

- **Se connecter :**
L'administrateur saisit son email et son mot de passe pour accéder à l'interface d'administration. Cette action déclenche une vérification côté backend via JWT.
- **Se déconnecter :**
L'administrateur peut volontairement mettre fin à sa session, ce qui efface le token d'authentification stocké localement.

Gestion des albums photo

- **Créer un album photo :**
Permet à l'administrateur de créer un nouvel album, en renseignant un nom. L'album est ensuite stocké côté base de données.
- **Modifier un album photo :**
Donne la possibilité de renommer un album existant.
- **Ajouter des photos :**
L'administrateur peut sélectionner et envoyer plusieurs photos dans un album. Les fichiers sont ensuite transférés sur Amazon S3 et liés à l'album via la base de données.
- **Supprimer des photos :**
Une ou plusieurs photos peuvent être supprimées d'un album. Cela supprime également les fichiers sur S3.
- **Supprimer un album photo :**
Cette action supprime l'album sélectionné ainsi que toutes les photos associées, dans la base de données et sur S3.

Gestion des documents

- **Ajouter un document :**
L'administrateur peut envoyer un document (PDF, Word, etc.), qui sera stocké sur Amazon S3 et enregistré dans la base.
- **Consulter les documents :**
Permet à l'administrateur de visualiser la liste des documents disponibles (nom, date, lien de téléchargement...).
- **Supprimer un document :**
Supprime un fichier du système (S3 et base de données).

Gestion de l'équipe

- **Créer un membre de l'équipe :**
Permet à l'administrateur d'ajouter une fiche membre, avec nom, prénom, rôle, et image de profil. Les données sont enregistrées en base et les images envoyées sur S3.
- **Modifier un membre de l'équipe :**
L'administrateur peut éditer les informations d'un membre existant (texte et/ou image).
- **Supprimer un membre de l'équipe :**
Supprime entièrement un membre du système et son image associée sur S3.

Chaque fonctionnalité est représentée comme un cas d'utilisation (ellipse), lié à l'acteur "Administrateur" par des associations. Ces cas sont regroupés selon leur domaine fonctionnel, facilitant la compréhension du périmètre fonctionnel de l'admin.

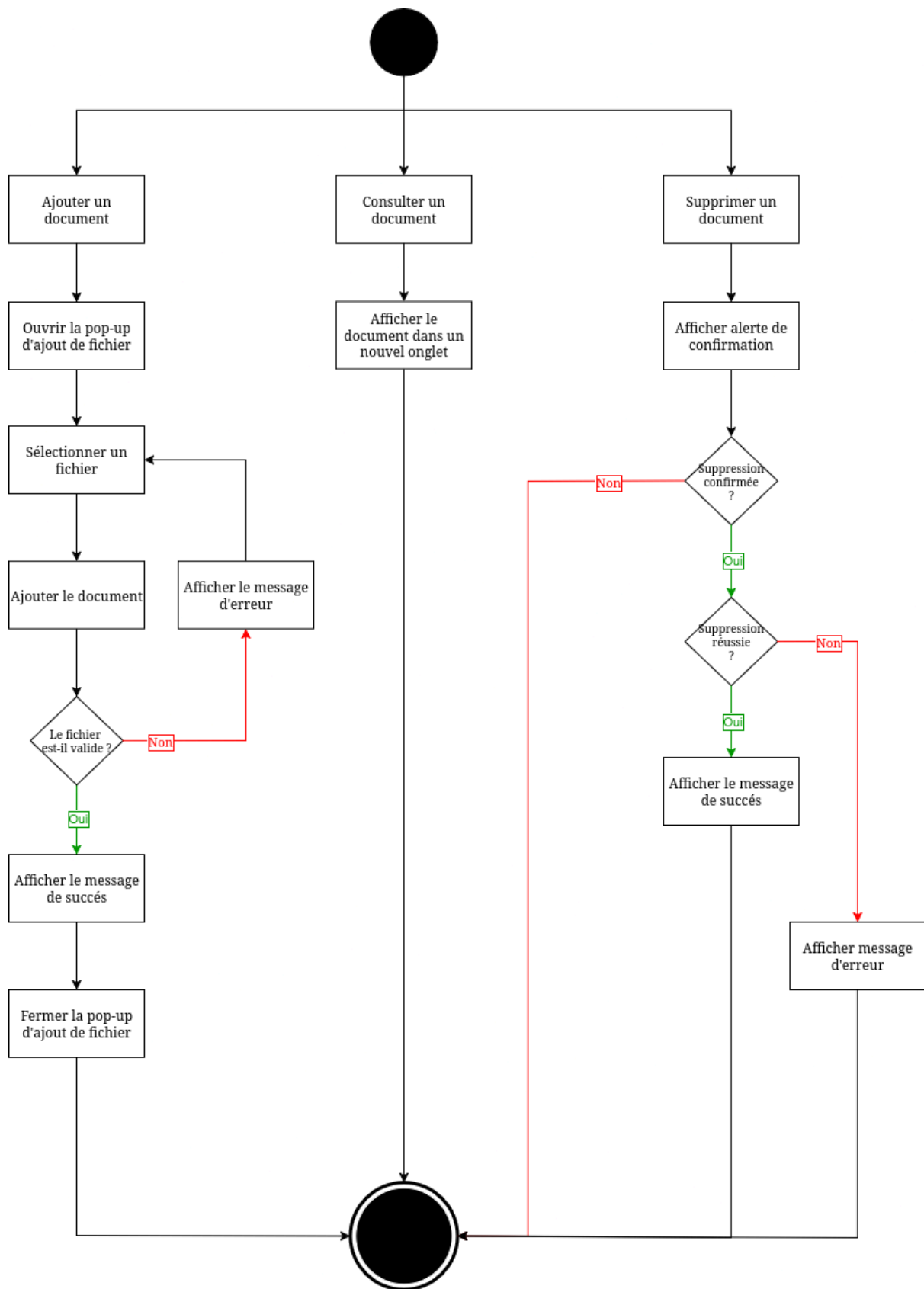
Justification

Ce diagramme permet de visualiser de façon claire les différentes fonctionnalités à implémenter côté administrateur. Il a été utilisé comme base pour découper les tâches pendant le développement et assurer la couverture fonctionnelle des besoins exprimés.

Diagramme de Flux pour la Gestion des Documents

Résumé

Ce diagramme de flux décrit les processus et les interactions nécessaires pour gérer les documents dans une application. Il est divisé en trois sections principales, chacune représentant une action distincte que les utilisateurs peuvent effectuer sur les documents : ajouter un document, consulter un document, et supprimer un document.



Explication

1. Ajouter un document :

- **Ouvrir la pop-up d'ajout de fichier** : L'utilisateur commence par ouvrir une interface pour ajouter un fichier.
- **Sélectionner un fichier** : L'utilisateur sélectionne un fichier à ajouter.
- **Ajouter le document** : Le fichier est ajouté au système.
- **Validation** : Le système vérifie si le fichier est valide.
 - Si le fichier est valide, un message de succès est affiché et la pop-up d'ajout de fichier est fermée.
 - Si le fichier n'est pas valide, un message d'erreur est affiché.

2. Consulter un document :

- **Afficher le document dans un nouvel onglet** : L'utilisateur peut consulter le document qui s'ouvre dans un nouvel onglet du navigateur.

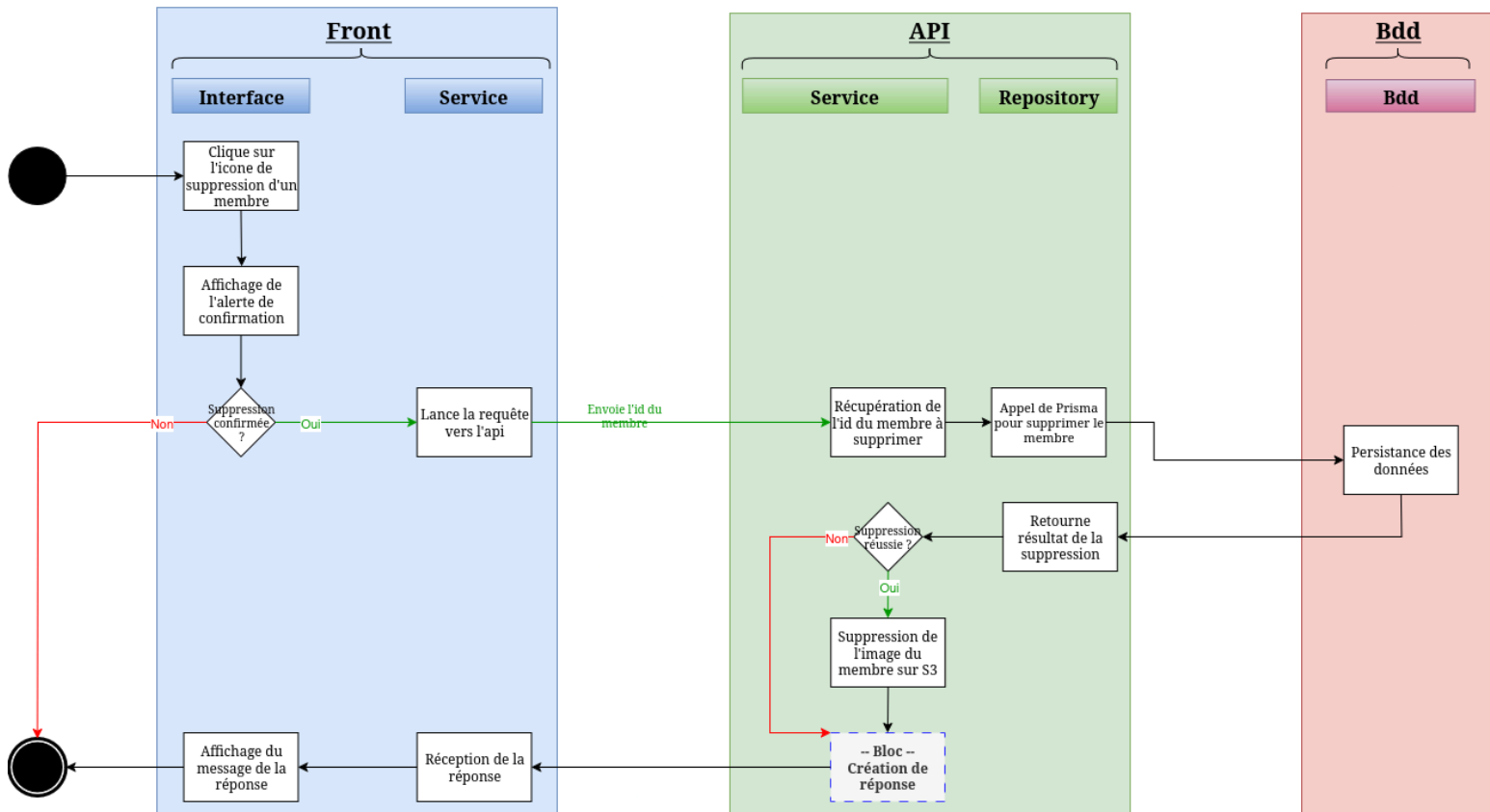
3. Supprimer un document :

- **Afficher alerte de confirmation** : L'utilisateur reçoit une alerte pour confirmer la suppression.
- **Confirmation de suppression** : L'utilisateur confirme ou annule la suppression.
 - Si la suppression est confirmée, le système vérifie si la suppression a réussi.
 - Si la suppression est réussie, un message de succès est affiché.
 - Si la suppression échoue, un message d'erreur est affiché.

Diagramme de Séquence pour la Suppression d'un Document

Résumé

Ce diagramme de séquence montre le flux détaillé des interactions entre les composants du système lors de la suppression d'un document. Il met en évidence les interactions entre le front-end, l'API, et la base de données.



Explication

1. Front-End

● Interface :

- **Clic sur l'icône de suppression d'un document** : L'utilisateur initie le processus de suppression en cliquant sur l'icône de suppression associée à un document.
- **Affichage de l'alerte de confirmation** : Une alerte de confirmation s'affiche pour demander à l'utilisateur de confirmer la suppression du document. Cela permet de s'assurer que l'utilisateur souhaite réellement supprimer le document et d'éviter les suppressions accidentelles.

● Service :

- **Lancement de la requête vers l'API** : Si l'utilisateur confirme la suppression, le service du front-end envoie une requête à l'API pour procéder à la suppression du document.

2. API

- **Service :**

- **Réception des données :** L'API reçoit les données de la requête envoyée par le front-end.
- **Vérification des données reçues :** L'API vérifie la validité des données reçues pour s'assurer qu'elles sont complètes et correctes.
 - **Données correctes :** Si les données sont valides, elles sont converties en DTO (Data Transfer Object) pour faciliter leur traitement et leur transfert entre les différentes couches de l'application.
 - **Appel de Prisma pour mettre à jour l'URL du document :** L'API utilise Prisma, un ORM (Object-Relational Mapping), pour mettre à jour l'URL du document dans la base de données. Cela peut inclure la suppression de l'URL ou sa mise à jour pour refléter l'état supprimé du document.
 - **Retour du résultat de l'opération :** L'API retourne le résultat de l'opération de suppression au front-end, indiquant si la suppression a été réussie ou non.

- **Repository :**

- **Interaction avec la base de données :** Le repository interagit directement avec la base de données pour effectuer les opérations de suppression et de mise à jour des données.

3. Base de Données (Bdd)

- **Bdd :**

- **Persistance des données :** La base de données assure la persistance des données mises à jour. Cela signifie que les modifications apportées aux données, telles que la suppression d'un document, sont enregistrées de manière permanente dans la base de données.

4. Retour au Front-End

- **Réception de la réponse :** Le front-end reçoit la réponse de l'API, indiquant si la suppression du document a été réussie ou non.
- **Affichage du message de réponse :** En fonction de la réponse reçue, le front-end affiche un message approprié à l'utilisateur.
 - **Message de succès :** Si la suppression a été réussie, un message de succès est affiché pour informer l'utilisateur que le document a été supprimé avec succès.
 - **Message d'erreur :** Si la suppression a échoué, un message d'erreur est affiché pour informer l'utilisateur de l'échec et lui permettre de prendre des mesures correctives si nécessaire.

Conception globale

La conception globale de l'application Krav Maga Spirit est structurée pour offrir une expérience utilisateur fluide et sécurisée, tout en assurant une gestion efficace des documents et des données. L'architecture de l'application est divisée en plusieurs couches distinctes, chacune ayant des responsabilités spécifiques, ce qui permet une séparation claire des préoccupations et une maintenance simplifiée.

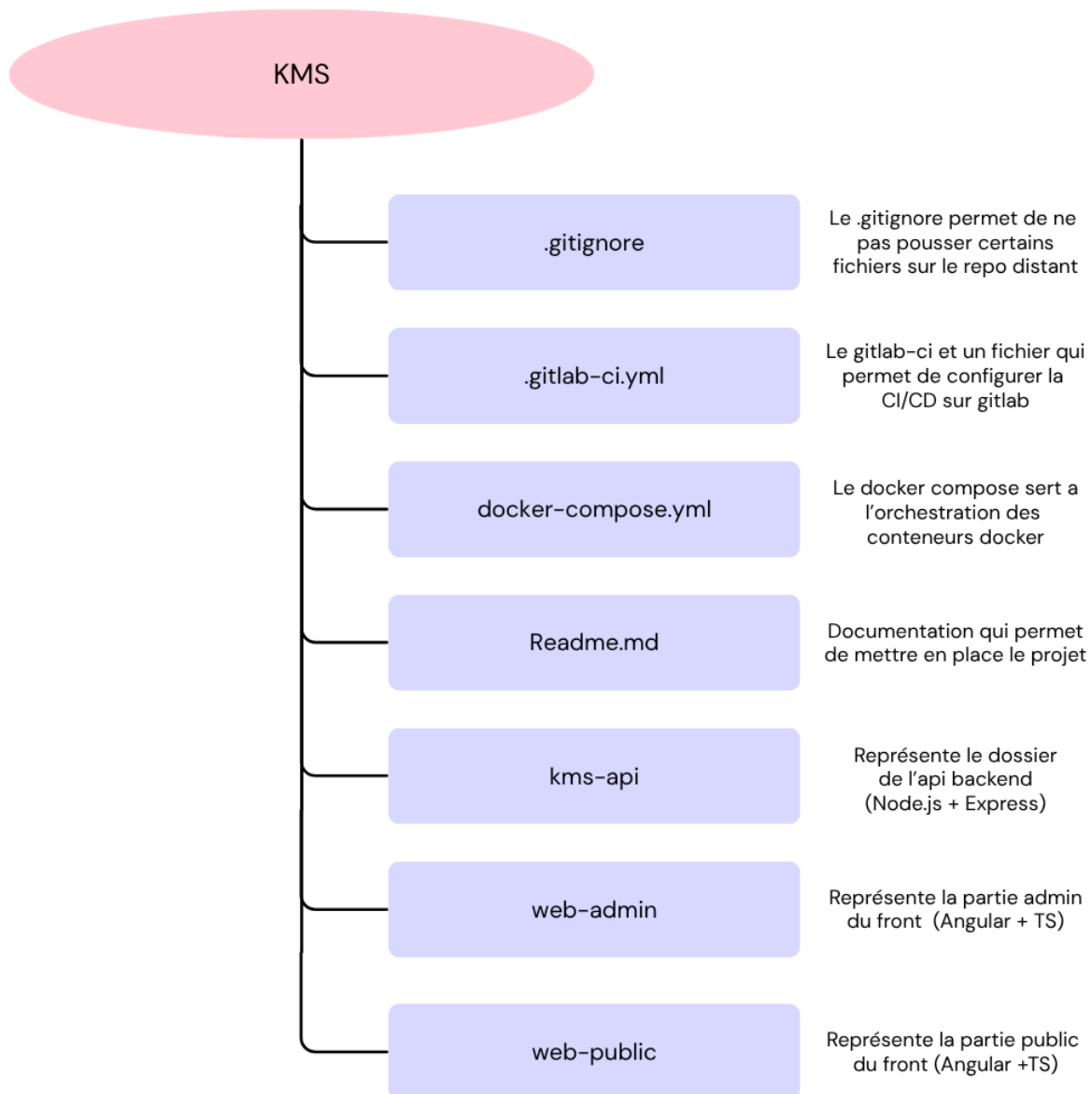
Le front-end est conçu pour être intuitif et réactif, offrant aux utilisateurs une interface conviviale pour interagir avec les fonctionnalités de l'application. Il est composé d'un site public accessible aux utilisateurs finaux et d'un panneau d'administration réservé aux administrateurs pour la gestion des utilisateurs et des contenus.

Le back-end est responsable de la logique métier et de la gestion des données. Il est composé d'une API qui expose des services pour le front-end, permettant ainsi la communication entre les différentes parties de l'application. L'API est conçue pour être robuste et sécurisée, assurant la validation des données et l'exécution des opérations de gestion des documents. Elle interagit avec une base de données pour la persistance des données, assurant ainsi leur intégrité et leur sécurité.

L'architecture globale de l'application est conçue pour être scalable et performante, capable de supporter les pics de fréquentation et d'offrir une expérience utilisateur fluide en permanence.

Présentation de la réalisation

Afin d'offrir une vue d'ensemble claire de la réalisation du projet, cette partie présente l'organisation générale de l'architecture applicative. Elle permet de comprendre comment les différentes composantes du code sont structurées, déployées et articulées entre elles pour former un système cohérent et fonctionnel.



L'arborescence complète du site est disponible dans les annexes de ce dossier afin de ne pas avoir un screen illisible.

L'arborescence du projet KMS s'aligne parfaitement avec les bonnes pratiques d'une architecture Angular modulaire et évolutive. La séparation claire entre le backend (api) et les deux interfaces frontend (web-admin et web-public) permet d'isoler les responsabilités et de faciliter les cycles de développement, de tests et de déploiement. Cette structure, compatible avec une approche microservices et adaptée à une gestion par conteneurs via Docker, offre une grande flexibilité pour faire évoluer l'application, ajouter de nouvelles fonctionnalités ou intégrer d'autres services dans le futur, sans compromettre la stabilité de l'ensemble.

Dans la continuité de cette présentation technique je vous présente les visuels pour vous faire découvrir les différentes interfaces du site, leur ergonomie, leur cohérence

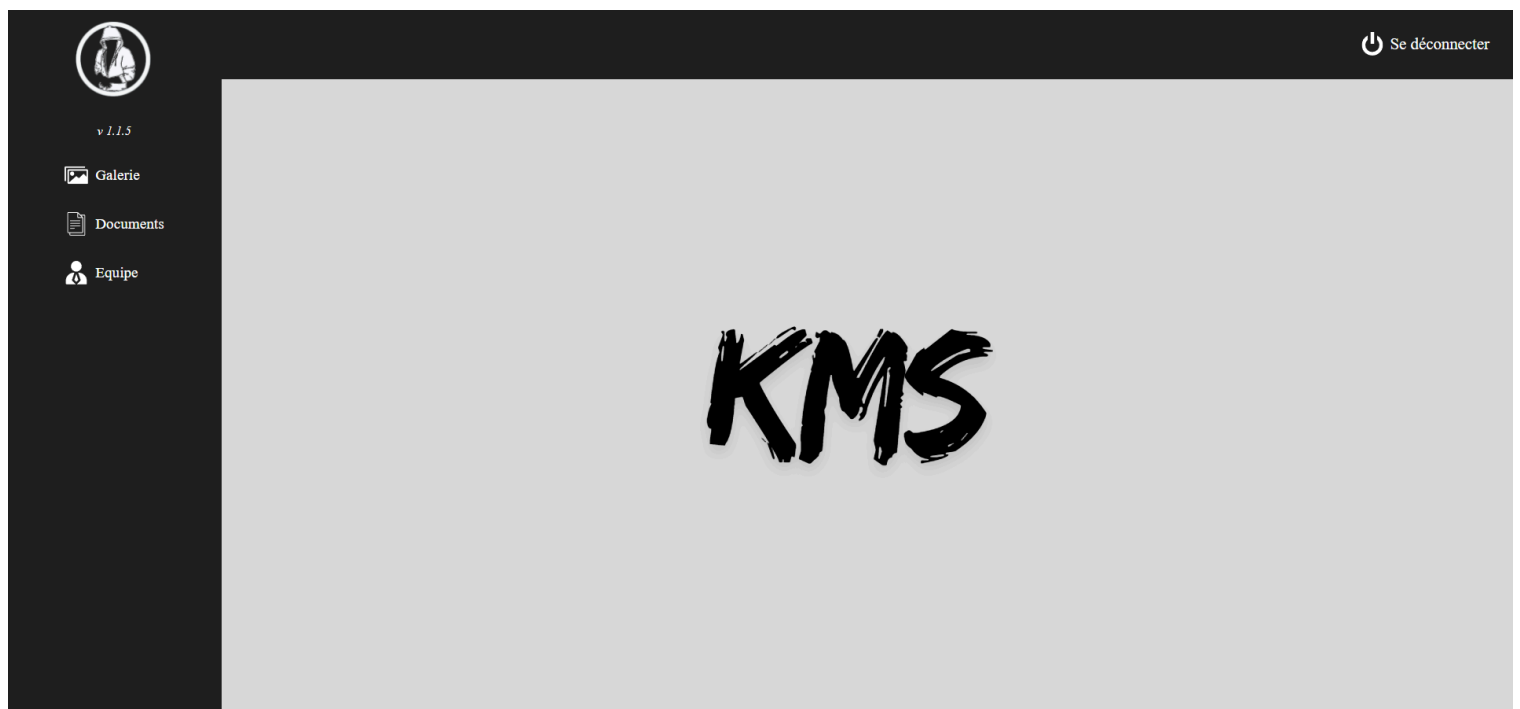
graphique, ainsi que les principales fonctionnalités accessibles aux utilisateurs et aux administrateurs.

La page de connexion constitue la porte d'entrée de l'interface d'administration du

A login form centered on a light gray background. The form is white with rounded corners and contains two input fields: 'Identifiant' (Username) and 'Mot de passe' (Password). Each field has a small eye icon to toggle visibility. Below the fields is a green button with the text 'Se connecter' (Log in).


site KMS. Elle a été conçue pour allier simplicité d'utilisation, sécurité et accessibilité. Son design est sobre et épuré. Accompagnés d'un système de gestion des erreurs en cas de saisie incorrecte. Côté sécurité, l'authentification repose sur un système de jetons JWT, garantissant un accès protégé à l'espace administrateur. Cette page a également été pensée pour être responsive, assurant une compatibilité optimale sur tous types d'écrans (ordinateur, tablette, mobile). Elle représente une étape essentielle du parcours utilisateur pour les responsables du club, leur donnant accès aux fonctionnalités de gestion de contenu.

Une fois connecté nous arrivons sur la page d'administration

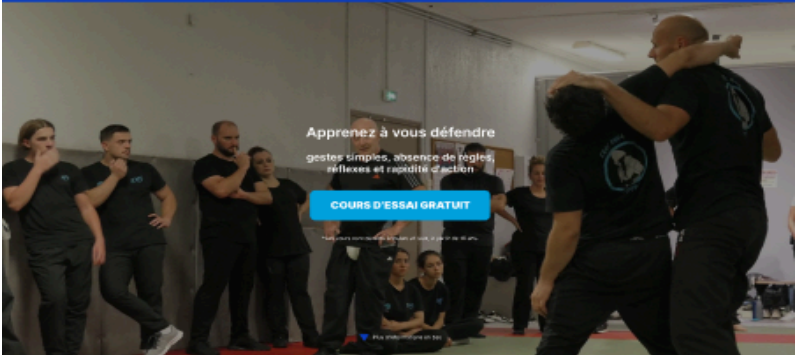


Une fois connecté, l'administrateur est redirigé vers la page d'accueil de l'interface d'administration. Cette page centrale joue le rôle de tableau de bord et offre une vue d'ensemble claire des principales fonctionnalités disponibles. Elle a été conçue pour faciliter la navigation et permettre un accès rapide aux modules essentiels : gestion de l'équipe, des photos, des documents. L'organisation visuelle repose sur une structure modulaire en cartes ou sections, chaque zone étant bien identifiée pour guider l'administrateur dans ses actions. L'ergonomie a été travaillée pour permettre une prise en main intuitive, même pour un utilisateur peu expérimenté, tout en garantissant une bonne expérience utilisateur grâce à la fluidité de l'interface Angular. Cette page constitue le point de départ de toutes les actions de gestion de contenu au sein du site.

Ensuite nous passons sur la partie public qui sera accessible par tout le monde sans besoins de se connecter


KRAV MAGA SPIRIT

[A PROPOS](#)
[KRAV MAGA](#)
[GALERIE](#)
[DOCUMENTS](#)
[CONTACT](#)



Apprenez à vous défendre
gestes simples, absence de règles
réflexes et rapidité d'action

COURS D'ESSAI GRATUIT

Maximum 10 personnes par séance - 1h30 de cours - 10€ de matériel

[Plus d'informations en bas](#)

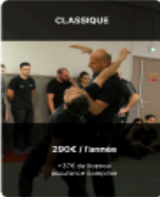
Horaires

LUNDI	VENDREDI
18h30 - 21h	20h - 21h30
Tous niveaux	Tous niveaux

Équipement nécessaire

- Pantalons + t-shirt noir
- Ceint de boxe ou gants ouvert
- Casque (Homme et Femme)
- Protection poitrine féminine
- Protège-tibia (conseillé)
- Chaussures à semelles souples (utilisées uniquement en intérieur)


Tarifs



CLASSIQUE

280€ / l'année

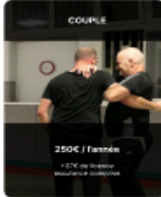
+ 17€ de matériel
Assurance + Coordonnée



ETUDIANTS

250€ / l'année

+ 17€ de matériel
Assurance + Coordonnée




COUPLE


350€ / l'année

+ 17€ de matériel
Assurance + Coordonnée

Localisation

Dojo Maison Pour Tous
70 avenue Zénatti, 13008 Marseille




KRAV MAGA SPIRIT

Maison Pour Tous

[Facebook](#)

© Krav Maga Spirit
70 avenue Zénatti, 13008 Marseille
Maison Pour Tous - 04 91 38 10 10

La partie publique du site, accessible à tous les visiteurs, a été pensée pour refléter l'identité du club Krav Maga Spirit. La page d'accueil joue un rôle clé dans cette première impression : elle présente les éléments essentiels du club (informations sur les cours, tarifs, emplacements...) de manière visuellement engageante. L'interface, développée en Angular, adopte une architecture Single Page Application (SPA) garantissant rapidité et réactivité lors de la navigation. Cette page a été construite pour donner les informations essentielles au client et permet aussi de naviguer sur les autres parties du site: A propos, Krav Maga, Galeries, Documents, Contact. Elle sert ainsi de vitrine principale du site et reflète le sérieux et le dynamisme du club.

Organisation du déploiement

Processus de mise en production (Déploiement)

Préparation du déploiement avec Docker Compose

La préparation du déploiement repose sur l'utilisation de Docker et de docker-compose, afin d'unifier et d'automatiser la gestion des différents services de l'application. Chaque composant du projet kms-api, front public, front-admin, base de données PostgreSQL MongoDB, microservice d'authentification, Traefik est défini dans un fichier docker-compose.yml qui permet de les lancer dans des conteneurs isolés. Cela garantit un environnement stable et identique en développement comme en production. Des fichiers spécifiques (comme les variables d'environnement) sont configurés pour adapter les comportements des services au contexte de production, assurant ainsi sécurité, performance et cohérence de l'ensemble de l'infrastructure.

Déploiement automatisé avec GitLab CI/CD

Le pipeline GitLab CI/CD joue un rôle central dans le déploiement automatisé de l'application. À chaque mise à jour du code dans la branche principale, GitLab exécute automatiquement un ensemble de tâches définies dans un fichier .gitlab-ci.yml. Cela inclut la construction des images Docker, les vérifications nécessaires (lint, build), et enfin le déploiement sur le serveur de production via SSH. Ce processus permet de réduire les erreurs humaines, d'assurer une traçabilité complète des versions et de déployer de nouvelles fonctionnalités de manière fluide et sécurisée.

Vérifications et actions post-déploiement

Une fois le déploiement terminé, des vérifications post-déploiement sont réalisées afin de s'assurer que les services sont bien opérationnels. Cela comprend la surveillance des logs des conteneurs, la vérification des accès aux différentes interfaces (admin et public), le bon fonctionnement des APIs, ainsi que la stabilité des connexions à la base de données. Des tests manuels complémentaires sont également effectués pour valider l'intégrité des principales fonctionnalités. Enfin, un

suivi est mis en place pour détecter rapidement d'éventuels dysfonctionnements après mise en ligne.

Tests et recette

Les tests font partie intégrante du cycle de développement de l'application afin de garantir la qualité, la fiabilité et la sécurité du système. Qu'il s'agisse de vérifications automatiques ou manuelles, ils permettent de détecter les anomalies le plus tôt possible, d'éviter les régressions, et de s'assurer que les fonctionnalités livrées répondent bien aux attentes définies. Dans ce projet, les tests ont été réalisés de manière itérative, à chaque étape de développement, en mettant l'accent sur la logique métier, la sécurité, et l'expérience utilisateur.

Phases de test et validation

Les phases de test se sont déroulées en parallèle du développement, avec plusieurs niveaux de vérification : tests unitaires sur les composants critiques, tests d'intégration pour s'assurer du bon fonctionnement entre les différentes parties (front, back, base de données), et tests manuels réalisés via l'interface utilisateur. Chaque fonctionnalité développée faisait l'objet d'une validation par au moins un autre membre de l'équipe. Des tests de non-régression ont également été réalisés à chaque nouvelle version, afin de vérifier que les ajouts n'impactent pas les fonctionnalités existantes.

Recettes de tests

Une fois l'ensemble des développements terminés, une phase de **recette** a été effectuée pour valider globalement l'application avant mise en production. Cette recette utilisateur a permis de tester l'application dans des conditions proches de la production, avec des cas d'usage réels et des comptes utilisateurs simulés. Des scénarios de tests prédéfinis ont été suivis (connexion, gestion des équipes, ajout de ressources, navigation front public, etc.). Les retours ont permis de corriger les derniers détails techniques ou ergonomiques. Cette étape a confirmé la conformité du livrable final par rapport aux objectifs fonctionnels initiaux.

Bilan projet :

Analyse Budgétaire

Introduction

L'analyse budgétaire permet d'évaluer les ressources mobilisées pour la réalisation du projet, tant du point de vue humain que matériel. Elle offre une vision claire des investissements nécessaires au bon déroulement du développement et à la mise en ligne de l'application, en prenant en compte les moyens techniques, humains et les outils utilisés.

Coûts humains

L'équipe de développement était composée de trois développeurs mobilisés de janvier à fin juillet. Chaque mois, une semaine de travail en présentiel était organisée (sauf en juillet, entièrement en présentiel). En partant d'une estimation conventionnelle de 100 € par jour de travail par développeur, et en considérant une semaine de 5 jours par mois pendant 6 mois (janvier à juin) puis 20 jours pour juillet, cela représente :

- **Janvier à juin** : $5 \text{ jours} \times 6 \text{ mois} \times 3 \text{ devs} \times 100 \text{ €} = \mathbf{9\,000 \text{ €}}$
- **Juillet** : $20 \text{ jours} \times 3 \text{ devs} \times 100 \text{ €} = \mathbf{6\,000 \text{ €}}$

Total estimé des coûts humains : 15 000 €

Outils utilisés

Divers outils ont été mis en place pour faciliter la gestion du projet, la communication, la conception et le développement. Ces outils, majoritairement gratuits ou déjà disponibles via des comptes personnels ou scolaires, incluent NeoVim pour l'édition de code, GitLab pour la gestion de version et le déploiement, Figma pour les maquettes, Google Drive et Google Docs pour le partage et la rédaction, Canva et Draw.io pour les visuels, ainsi que des outils de communication comme Google Chat. Leur utilisation a permis de limiter les coûts supplémentaires tout en assurant un bon suivi et une bonne collaboration.

Coûts d'infrastructure

Concernant les infrastructures techniques, le projet repose sur des services en ligne impliquant des frais mensuels fixes :

- Hébergement OVH : $15 \text{ €/mois} \times 12 \text{ mois} = \mathbf{180 \text{ €}}$
- Stockage Amazon S3 : $5 \text{ €/mois} \times 12 \text{ mois} = \mathbf{60 \text{ €}}$

- Nom de domaine et certificat ssl : **20 €/ans**

Total des coûts d'infrastructure : 260 €/ans

Ce qui fait un cout totale d'infrastructures pour le client après déploiement de 260 euros par an ce qui représente une dépense moindre et bien gérée.

Coût total du projet

En combinant les coûts humains (15 000 €) et les coûts d'infrastructure (280 €), le **coût total estimé du projet s'élève à 15 280 €**.

Cette estimation reflète l'engagement de l'équipe et l'optimisation des ressources dans la réalisation d'un projet web complet et professionnel, avec une architecture technique robuste, un design fonctionnel et un déploiement maîtrisé.

Résultat atteints

L'application a été développée avec une stack moderne : Angular, Node.js et PostgreSQL. Nous avons fait face à quelques défis avec l'intégration d'Amazon S3. L'équipe s'est bien adaptée au projet dans l'ensemble que cela soit au niveau des technos ou au niveau de l'environnement de travail.

La mise en prod sur le VPS OVH sera finalisée sous peu, tout fonctionne correctement : API, interfaces, base de données et Traefik. L'architecture avec Docker assure la stabilité et la sécurité. Le CI/CD GitLab est en place pour automatiser les futurs déploiements.

Les deux interfaces répondent bien aux besoins du club. Le site public a un look moderne et l'interface admin permet aux responsables de gérer le contenu facilement. Les retours lors des tests avec le club confirment que c'est beaucoup mieux que l'ancien site, surtout pour la gestion des photos.

Au 1er août 2025, le projet KMS pour le club Krav Maga Spirit est bouclé. La phase de conception avait bien posé les bases avec une architecture claire et évolutive, ce qui nous a permis de développer une app solide qui correspond aux besoins du club.

L'API REST avec Node.js/Express.js et Prisma ORM fait bien le lien entre les interfaces et la base PostgreSQL. Ça assure de bonnes performances et la sécurité des données.

On a fait une série de tests complets pour vérifier que tout marche bien et que ça respecte les demandes du club. L'authentification admin est sécurisée et l'app est prête pour la mise en service.

La formation des admins du club est prévue pour le 26 août avec toute la doc et les sessions de prise en main.

L'architecture qu'on a mise en place permet d'ajouter facilement de nouvelles fonctionnalités plus tard selon les besoins. On attend maintenant les retours des utilisateurs en conditions réelles pour voir s'il faut faire des ajustements et continuer à améliorer la plateforme.

Bilan personnel :

Retours d'expériences

Le projet KMS s'est globalement bien déroulé, même s'il n'a pas été exempt de difficultés. Il a représenté un véritable défi à plusieurs niveaux, notamment en raison de la courbe d'apprentissage importante que nous avons dû franchir. Pour une grande partie de l'équipe, c'était la première fois que nous avions à gérer un projet complet à plusieurs, avec tout ce que cela implique en matière de coordination, de communication et de partage des responsabilités.

La gestion du temps a été un point particulièrement complexe à maîtriser. Le rythme de l'alternance, avec une présence en entreprise la majeure partie du temps et peu de semaines de regroupement complet, a nécessité une organisation rigoureuse. Nous avons dû apprendre à respecter une méthodologie de travail commune, à répartir les tâches et à planifier notre avancement malgré un contexte morcelé. Travailler en équipe dans ces conditions nous a appris à faire preuve d'autonomie tout en maintenant une cohésion nécessaire pour avancer ensemble.

Le fait d'avoir un client réel avec des attentes concrètes a aussi représenté un tournant. Il a fallu adapter notre discours, comprendre les besoins exprimés parfois de manière non technique, et savoir formaliser les demandes en tâches techniques compréhensibles pour tous. Cela a renforcé notre sens des priorités et notre capacité à faire des compromis réalistes entre attentes du client et faisabilité technique dans le temps imparti.

Techniquement, ce projet nous a permis de toucher à de nombreuses notions avancées. La gestion d'un environnement de production avec un VPS, l'intégration de Traefik pour le routing dynamique des services, ou encore l'orchestration avec Docker nous ont permis de mettre en place une infrastructure stable et moderne. Ces éléments, nouveaux pour la plupart d'entre nous, ont demandé un effort d'apprentissage conséquent, mais ont été très enrichissants.

Sur le plan du développement, la montée en compétences sur Angular et Node.js a été très significative. Nous avons appris à structurer une application front moderne, à utiliser les composants, les services, la gestion d'état, et à faire dialoguer le tout avec une API construite sur Express.js. Le choix de Prisma comme ORM a également été un point clé pour garantir une bonne gestion de notre base de données tout en profitant de la sécurité apportée par le typage TypeScript.

Le travail en équipe a nécessité une vraie rigueur dans la gestion de Git. Nous avons mis en place une stratégie de branches claire, des revues de code, et appris à résoudre des conflits efficacement. Par ailleurs, nous avons intégré progressivement une logique de tests pour valider notre code, ainsi qu'un pipeline CI/CD avec GitLab, afin d'automatiser les déploiements et les vérifications de qualité. Cette étape nous a permis de mieux comprendre les enjeux de l'intégration continue et de la livraison automatisée dans un contexte professionnel.

Projet professionnel

Bilan de l'expérience

Je suis très satisfait de ces derniers mois consacrés à la création et à l'accompagnement de ce projet. Cette expérience s'inscrit dans un parcours de développement professionnel que j'ai entamé il y a maintenant trois ans, lorsque j'ai intégré la formation Start à La Plateforme, puis rejoint l'entreprise Cityway en tant que développeur alternant.

Objectifs à court terme

Dans l'immédiat, mon objectif principal est de finaliser et de réussir mon année en cours. Cette période représente une étape cruciale dans mon parcours de formation et je souhaite consolider les compétences acquises pour poser des bases solides pour la suite de ma carrière.

Perspectives à moyen terme

À moyen terme, j'envisage de poursuivre mon développement professionnel en intégrant un master au sein de l'école Ynov et l'entreprise Cityway. Cette formation me permettra de monter en compétence sur deux années supplémentaires et d'approfondir mes connaissances techniques. Parallèlement à cette formation, je projette de développer une application personnelle que j'aimerais mettre en production, ce qui constituerait un excellent complément pratique à mon cursus théorique.

Vision à long terme

Sur le long terme, mon ambition est de devenir un développeur expert capable de mener à bien des missions variées et complexes. Je souhaite développer une polyvalence technique qui me permettra d'aborder différents types de projets avec aisance et efficacité. Cette expertise me donnerait la possibilité de contribuer significativement aux projets d'entreprise et d'évoluer vers des responsabilités plus importantes.

Annexe :

Afin de ne pas prendre trop de place et gardé des document lisible les annexes seront disponible via les liens suivant.

Documentation

[Cahier des charges](#) : Document initial définissant les besoins fonctionnels et non fonctionnels du client.

[Spécifications](#) : Détail technique des fonctionnalités attendues à partir du cahier des charges.

[User stories](#) : Description des fonctionnalités sous forme de scénarios utilisateur pour guider le développement.

Graphiques

- Charte graphique

[Panel admin](#) : Charte graphique de l'interface dédiée aux administrateurs du site, présentant les fonctionnalités de CRUD.

[Site public](#) : Charte graphique de l'interface dédiée aux public, présentant le club dans sa globalité.

- Maquettes public

[Accueil](#) : Maquettes wireframe et propre de la partie public du site web représentant la page d'accueil.

[A propos](#) : Maquettes wireframe et propre de la partie public du site web représentant la page a propos.

[Krav maga](#) : Maquettes wireframe et propre de la partie public du site web représentant la page d'histoire du krav maga.

[Galerie](#) : Maquettes wireframe et propre de la partie public du site web représentant la page galerie.

[Documents](#) : Maquettes wireframe et propre de la partie public du site web représentant la page documents.

[Contact](#) : Maquettes wireframe et propre de la partie public du site web représentant la page de contact.

- Maquettes administrateur

[Authentification](#) : Maquettes wireframe et propre de la partie administrateur du site web représentant la page d'authentification.

[Accueil](#) : Maquettes wireframe et propre de la partie administrateur du site web représentant la page d'accueil.

[Documents](#) : Maquettes wireframe et propre de la partie administrateur du site web représentant la page documents.

[Galerie](#) : Maquettes wireframe et propre de la partie administrateur du site web représentant la page galerie.

[Équipes](#) : Maquettes wireframe et propre de la partie administrateur du site web représentant la page équipes.

Base de données

- PostgreSQL

[MCD](#) : Modèle conceptuel de données de la base de données relationnelle

[MLD](#) : Modèle logique de données de la base de données relationnelle

[MPD](#) : Modèle physique de données de la base de données relationnelle

- MongoDB

[MCD](#) : Modèle conceptuel de données de la base de données non relationnelle

[MLD](#) : Modèle logique de données de la base de données non relationnelle

[MPD](#) : Modèle physique de données de la base de données non relationnelle

Diagrammes

- Diagrammes classes

[Administrateur](#) : Représente les objets métiers manipulés dans la partie admin (classes, attributs, relations).

[API - Album](#) : Décrit les structures de données propres à l'API album.

[API - Authentification](#) : Décrit les structures de données propres à l'API authentification.

[API - Contact](#) : Décrit les structures de données propres à l'API contact.

[API - Document](#) : Décrit les structures de données propres à l'API document.

[API - Équipe](#) : Décrit les structures de données propres à l'API equipe.

[API - Upload](#) : Décrit les structures de données propres à l'API upload.

[Public](#) : Représente les objets métiers manipulés dans la partie public (classes, attributs, relations).

- Diagrammes d'activités

[Administrateur authentification](#) : Illustrent les différentes étapes du processus métier de l'authentification côté admin.

[Administrateur documents](#) : Illustrent les différentes étapes du processus métier du documents côté admin.

[Administrateur équipe](#) : Illustrent les différentes étapes du processus métier de l'équipe côté admin.

[Administrateur contact](#) : Illustrent les différentes étapes du processus métier du documents côté contact.

[Public Contact](#) : Schéma représentant la soumission d'un message depuis le formulaire de contact public.

- Diagrammes cas d'utilisation

[Administrateur](#) : Représentation des actions possibles pour un administrateur avec ses interactions principales avec le système.

[Public](#) : Représentation des actions possibles pour un utilisateur avec ses interactions principales avec le système.

- Diagrammes de séquences

[Album création](#) : Représentation graphique des interactions entre les acteurs et le système pour la création d'album.

[Album modification](#) : Représentation graphique des interactions entre les acteurs et le système pour la modification d'album.

[Album suppression](#) : Représentation graphique des interactions entre les acteurs et le système pour la suppression d'album.

[Authentification](#) : Représentation graphique des interactions entre les acteurs et le système pour l'authentification.

[Bloc - création de réponse](#) : Représentation graphique des interactions entre les acteurs et le système pour la création d'un bloc de réponse.

[Ajout de document](#) : Représentation graphique des interactions entre les acteurs et le système pour l'ajout d'un document.

[Consultation de document](#) : Représentation graphique des interactions entre les acteurs et le système pour la consultation d'un document.

[Suppression de document](#) : Représentation graphique des interactions entre les acteurs et le système pour la suppression d'un document.

[Création de membre](#) : Représentation graphique des interactions entre les acteurs et le système pour la création de membre.

[Modification de membre](#) : Représentation graphique des interactions entre les acteurs et le système pour la modification de membre.

[Suppression d'un membre](#) : Représentation graphique des interactions entre les acteurs et le système pour la suppression d'un membre.

Autres

[Arborescence de l'api](#) : Représentation de l'arborescence côté API.

[Arborescence de l'administrateur](#) : Représentation de l'arborescence côté administrateur.

[Arborescence du projet](#) : Représentation de l'arborescence projet.

[Arborescence du public](#) : Représentation de l'arborescence côté public.

[Architecture application en production](#) : Représentation de l'architecture de l'application une fois déployé

[Architecture application](#) : Représentation de l'architecture de l'application en général

[Diagramme de Gantt](#) : Calendrier de prévision pour le déroulement du projet

[Git exemple branches](#) : Exemple de versionning git.

[Gitlab issues board](#) : Exemple du issues board utilisés pour les tickets du projet présentés

[Gitlab exemple ticket](#) : Exemple d'une rédaction de ticket

Glossaire

API (Application Programming Interface) : Interface permettant à deux logiciels de communiquer entre eux. Utilisée ici pour faire dialoguer le front-end Angular et le

back-end Node.js.

Angular : Framework JavaScript côté client développé par Google, utilisé pour construire l'interface utilisateur du projet.

Authentification : Processus de vérification de l'identité d'un utilisateur avant de lui donner accès à des fonctionnalités sécurisées.

Back-end : Partie serveur de l'application, gère les données, la logique métier et la sécurité.

Back-office : Espace d'administration réservé aux membres du bureau, permettant la gestion du contenu du site (documents, membres, événements, etc.).

Base de données relationnelle : Système permettant de stocker des données organisées en tables liées entre elles, ici manipulées avec Prisma.

Bootstrap : Framework CSS utilisé dans la maquette Figma, facilitant la création d'interfaces responsive.

CI/CD (Intégration et Déploiement Continus) : Ensemble de pratiques DevOps visant à automatiser les tests, le build et le déploiement du code.

CMS (Content Management System) : Système de gestion de contenu. Dans le contexte du projet, un back-office personnalisé a été préféré à un CMS existant.

CRUD : Acronyme de Create, Read, Update, Delete – opérations de base d'une application web manipulant des données.

Docker : Plateforme de conteneurisation permettant de créer, déployer et exécuter des applications dans des environnements isolés et reproductibles.

Dépôt Git : Espace versionné sur GitLab permettant de stocker le code source et suivre les modifications.

Figma : Outil collaboratif de design d'interface utilisé pour la conception des maquettes du site.

Front-end : Partie visible de l'application web, ici développée avec Angular.

GitLab : Plateforme DevOps combinant gestion de code, suivi des tickets, CI/CD et documentation.

HTTPS (Hypertext Transfer Protocol Secure) : Version sécurisée du protocole HTTP utilisant un chiffrement TLS/SSL.

IAM (Identity and Access Management) : Système de gestion des droits et identités utilisé chez Cityway, permettant de centraliser les accès utilisateurs.

JWT (JSON Web Token) : Format de jeton d'authentification sécurisé utilisé pour les sessions utilisateurs.

Krav Maga Spirit (KMS) : Club de self-défense commanditaire du projet. Le site vise à mettre en valeur l'association et faciliter sa gestion interne.

Login/Logout : Fonctionnalités permettant aux utilisateurs de se connecter (login) et de se déconnecter (logout) de la plateforme.

Maquette : Prototype visuel statique du site utilisé pour valider l'ergonomie et le design avant développement.

MaaS (Mobility as a Service) : Modèle de mobilité intégrée développé chez Cityway, facilitant les transports multimodaux via une seule plateforme.

Node.js : Environnement d'exécution JavaScript côté serveur, utilisé pour construire l'API REST du projet.

ORM (Object Relational Mapping) : Technique permettant de manipuler une base de données via du code orienté objet. Ici, l'ORM utilisé est Prisma.

Prisma : ORM TypeScript utilisé pour interagir avec la base de données relationnelle du projet.

Power BI : Outil de visualisation de données utilisé par Cityway pour les tableaux de bord statistiques.

Responsive Design : Conception d'interfaces qui s'adaptent automatiquement à la taille de l'écran (ordinateur, tablette, smartphone).

Reverse Proxy : Serveur intermédiaire (comme Traefik) qui redirige les requêtes entrantes vers les bons services backend.

Scrum : Méthodologie Agile basée sur des cycles courts (sprints) et une planification collaborative.

Sprint : Période de développement définie (souvent 1 à 2 semaines) à l'issue de laquelle une fonctionnalité doit être livrée.

SSO (Single Sign-On) : Méthode d'authentification unique permettant à un utilisateur d'accéder à plusieurs services avec une seule connexion.

Traefik : Reverse proxy open-source utilisé pour le routage des services Docker via des règles dynamiques.

Token : Jeton d'authentification (souvent JWT) envoyé au client après login pour sécuriser les requêtes.

UX (User Experience) : Ensemble des éléments participant à la qualité de l'expérience vécue par l'utilisateur sur le site.

UI (User Interface) : Interface utilisateur, c'est-à-dire l'apparence visuelle et interactive du site web.

Versionnage : Suivi des différentes versions du code source grâce à Git.

VPS (Virtual Private Server) : Serveur virtuel dédié utilisé pour héberger l'application en production.

Web vitrine : Site internet présentant une entité (entreprise, association...) et ses services sans fonctionnalité de commerce en ligne.