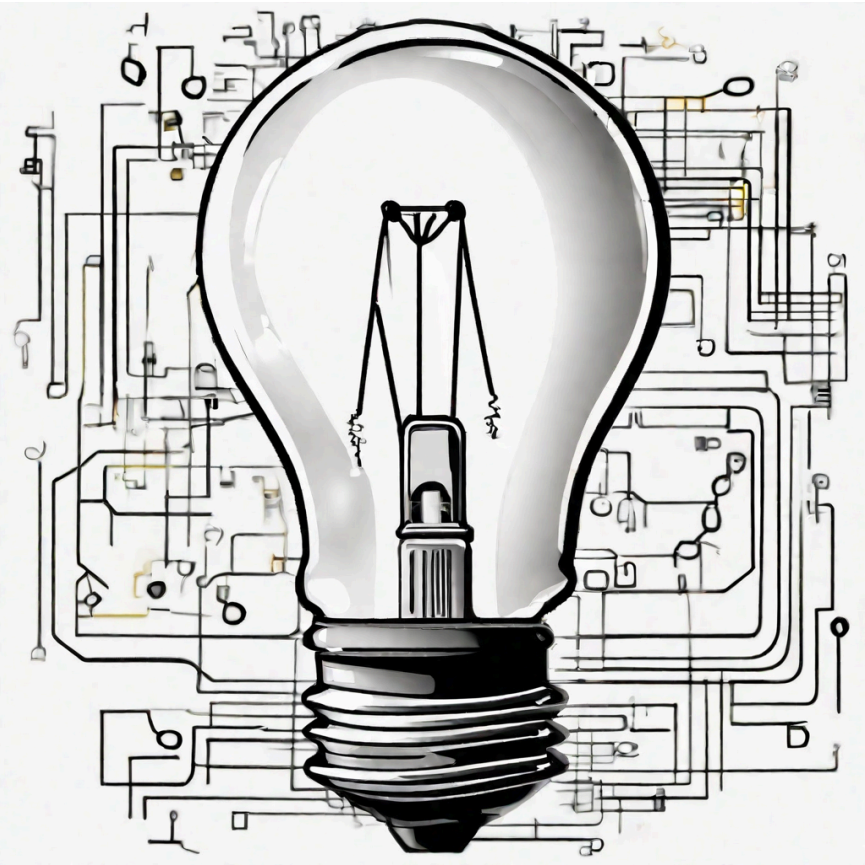
Projet de fin d'étude



Yanis Vroland

M2 Cyber

2024

1. Sommaire

2. Introduction

Dans le monde professionnel aujourd'hui, l'utilisation des données en entreprise, également connue sous le nom de "data-driven decision making" (prise de décision basée sur les données), est devenue une pratique des plus importantes. Les données sont une ressource stratégique non négligeable, et les entreprises qui parviennent à les collecter et à les analyser ont un avantage concurrentiel.

Dans cette univers de data, l'utilisation des informations liées aux clients et plus précisément aux projets, sont encore grandement sous-estimée. Les pratiques qui consistent à recueillir, classer et stocker de manière systématique toutes les informations liées aux projets réalisés par une organisation sont très peu utilisées. Pourquoi ça ? Car le faire soi- même prend du temps, les outils disponibles sont parfois trop coûteux et/ou pas adaptés aux besoins spécifiques des entreprises.

Ses informations non utilisées pourraient aider à démarrer de nouveaux projets rapidement mais également d'analyser les dépendances entre eux. De faciliter la coordination interne, la réduction des redondances, et favoriser une gestion optimale des ressources.

Ce rapport à pour but de suivre le développement d'une solution logiciel qui permet de répondre à une des problématiques que peut représenter le répertoriage des informations projets. Elle répondra plus précisément à : **Comment répertorier et rendre accessible les informations projets de manière spécifique à tout type d'entreprise ?** Cette question donne des interrogations sur les méthodes, les outils et les meilleures pratiques à adopter pour assurer un répertoriage pertinent.

L'objectif principal de ce projet est de conceptualiser et de mettre en œuvre une solution logicielle qui répond de manière efficace aux défis du marché concurrentiel. Dans ce rapport, il y a une partie sur les concepts clés du répertoriage de projet, en tenant compte des exigences spécifiques du marché, suivie des explications sur les méthodologies et l'organisation choisies pour guider le développement de la solution. On trouvera ensuite une partie sur un cahier des charges détaillant les besoins concrets du projet, les exigences fonctionnelles, ainsi que les contraintes et objectifs à atteindre. Il sera suivi de la partie conception qui comprend une modélisation de la solution logicielle ainsi que la partie réalisation qui nous parle du développement du logiciel. En fin de ce rapport, une partie dédiée à la validation, aux tests et à l'évaluation de la solution clôturera le processus de développement.

En résumé, ce rapport donnera une vision globale du processus de développement de la solution logicielle, en détaillant les aspects méthodologiques, conceptuels et techniques qui ont guidé la démarche.

Toutes les réalisations liées à ce projet sont accessibles via les liens suivants :

- Github de l'application mobile : https://github.com/YanisVroland/Mobile_MyTechLib
- Github de l'API : https://github.com/YanisVroland/API_MyTechLib

3. Étude de l'existant

Le répertoriage des informations en entreprise comprend différentes idées qui influent sur sa portée, sa méthodologie et son impact sur la gestion entreprise. Cette section parle de ces concepts pour donner les bases nécessaires à la compréhension du sujet.

3.1. Concepts

Le répertoriage des projets ne se limite pas à la simple organisation de données. Il s'agit d'une pratique systématique visant à rassembler, classifier et stocker de manière structurée toutes les informations pertinentes associées aux projets d'une organisation. Cela va au-delà du simple archivage, donnant une vision des projets antérieurs et en cours.

À ses débuts, la logique principale était mise sur la conservation et l'organisation des informations liées aux projets entrepris et non ceux déjà réalisés. Les entreprises ont reconnu la nécessité de documenter leurs projets pour en tirer des enseignements et favoriser une meilleure gestion. Cette première phase de répertoriage était principalement axée sur la documentation des projets en cours, permettant aux entreprises de retracer leur évolution, d'identifier les bonnes pratiques et de prévenir la répétition d'erreurs passées.

Au fil des années, les technologies appliquées au répertoriage des projets évoluent rapidement. Les solutions basées sur le cloud gagnent en popularité, offrant une accessibilité accrue, une flexibilité et une sécurité renforcée. Aujourd'hui l'intégration de l'intelligence artificielle (IA) dans les systèmes de répertoriage commence à émerger, permettant une analyse plus poussée des données, une prédiction des tendances et une automatisation accrue des tâches répétitives. L'expansion des technologies sur mobiles influence les solutions de répertoriage. Elles permettent aux utilisateurs d'accéder aux informations projet en temps réel, quel que soit leur emplacement géographique.

3.2. Méthodes et techniques utilisées

Dans le domaine du répertoriage des projets, deux approches principales se distinguent :

Les méthodes dites 'manuelles' impliquent souvent l'utilisation de feuilles de calcul, de documents physiques, ou de bases de données administrées manuellement par les équipes projet. Ces approches nécessitent une implication humaine pour collecter, classer et mettre à jour les informations. Les méthodes manuelles offrent une flexibilité dans la collecte d'informations mais peuvent être sensibles à des erreurs humaines, des retards, et à une mise à jour loupée des données.

D'un autre côté, les méthodes 'automatisées' améliorent l'efficacité, réduisent les erreurs, mais peuvent nécessiter une certaine adaptation et une maintenance continue. Elles reposent sur des outils logiciels spécialement conçus pour gérer le répertoire des projets. Ces outils peuvent automatiser la collecte de données, générer des rapports en temps réel, et offrir une visibilité accrue grâce à des fonctionnalités de recherche avancées. Par

exemple, des solutions de gestion de projet telles que Jira, Trello, ou Microsoft Project peuvent être utilisées pour automatiser la collecte de données, générer des rapports en temps réel et offrir une visibilité augmentée grâce à des fonctionnalités de recherche avancées.

Ces outils permettent de centraliser les informations liées aux projets, de suivre les avancements, et de collaborer efficacement au sein des équipes. Les avantages des méthodes automatisées incluant une gestion plus rapide et précise des données et une meilleure intégration avec d'autres systèmes d'entreprise. Mais, les coûts et la nécessité d'une formation ne sont pas à oublier.

3.3. Étude de marché

Dans le domaine du catalogage/répertoriage de projet, plusieurs acteurs, concurrents connus offrent des solutions spécialisées déjà existantes :

- Microsoft Project Server est reconnu comme l'une des principales solutions de gestion de projet. Cette plateforme permet le catalogage, la planification et le suivi précis des projets, tout en donnant des fonctionnalités avancées pour la gestion des ressources et la collaboration d'équipe.
- Primavera P6, développé par Oracle, se distingue dans les industries où la gestion de projet est plus complexe et exigeante. Il a des fonctionnalités avancées pour la planification et la gestion des projets de construction, d'ingénierie, et autres.
- Assembla, une plateforme de gestion de projet et de développement de logiciels qui offre aussi des solutions de catalogage de projet. En plus du catalogage, Assembla propose des fonctionnalités de suivi des tâches, de gestion des versions et de collaboration d'équipe.
- Redmine, une application open-source de gestion de projet, qui est utilisée dans les environnements de développement logiciel. Redmine donne des fonctionnalités de répertoriage de projet, de suivi des problèmes, de gestion des versions et de collaboration d'équipe, répondant ainsi aux besoins des équipes travaillant sur des projets logiciels complexes.

Microsoft Project Server et Primavera P6 dominent le marché de la gestion de projet, chacun avec ses forces : le premier bénéficie du soutien et de la fiabilité de Microsoft, tandis que le second excelle dans les environnements de gestion de projet complexes, notamment dans l'ingénierie. Assembla, bien que moins répandu, propose une plateforme polyvalente pour la gestion de projet et le développement logiciel. Enfin, Redmine, en tant qu'application open-source, a une forte communauté dans les environnements de développement logiciel, offrant des solutions flexibles et personnalisables.

Les entreprises ont plusieurs critères et des besoins bien précis. Parmi eux, la convivialité de l'interface utilisateur se positionne en tant que critère principal. Les entreprises recherchent des solutions intuitives, faciles à prendre en main, et capables de favoriser une adoption rapide par leurs équipes. Les entreprises rencontrent divers défis liés à la gestion de projets, tels que la complexité des projets, la gestion des ressources limitées et la coordination des équipes. Elles recherchent des solutions offrant une gestion agile et une

visibilité en temps réel sur l'avancement des projets. La minimisation des risques, la maximisation de la rentabilité et la réduction des redondances sont des objectifs clés pour optimiser l'efficacité opérationnelle. Certains secteurs, comme la santé, accordent une priorité absolue à la sécurité des données, tandis que le secteur financier met l'accent sur la conformité réglementaire. Les entreprises axées sur la recherche et le développement ont des besoins spécifiques en termes de gestion des ressources et de suivi des coûts.

Avec l'évolution rapide des technologies, les entreprises recherchent des solutions de répertoriage qui s'intègrent aux nouvelles avancées. L'intégration d'intelligence artificielle pour des analyses prédictives, la compatibilité avec les technologies blockchain pour une sécurité accrue, et l'utilisation de l'apprentissage automatique pour améliorer la classification des projets sont de plus en plus considérées comme des atouts.

Des lacunes dans les offres actuelles du marché des répertoires de projet suggèrent des opportunités d'innovation. L'intégration des technologies émergentes comme l'intelligence artificielle et la blockchain est une lacune majeure à exploiter, offrant aux entreprises des capacités d'analyse prédictive, une sécurité renforcée des données et une traçabilité transparente des informations de projet. Simplifier l'expérience utilisateur en créant des interfaces intuitives et permettant une personnalisation approfondie des solutions sont d'autres axes d'innovation. Améliorer la sécurité des données via des systèmes avancés de cryptage et d'authentification peut renforcer la confiance et garantir l'intégrité des données et la confidentialité des informations sensibles.

En conclusion de cette partie, on sait que l'évolution des technologies, notamment l'intégration du cloud, de l'intelligence artificielle, et des solutions mobiles, apporte des changements significatifs dans les possibilités du répertoriage de projets. Les méthodes automatisées offrent des avantages en termes de gestion rapide et précise des données, même si c'est accompagné de coûts et d'un besoin de formation.
L'étude de marché met en avant les critères pour les entreprises dans le domaine du répertoriage des projets, allant de la partie interface utilisateur conviviale à la nécessité de minimiser les risques et de personnaliser les solutions. Les opportunités d'innovation repérées comprennent aussi une intégration plus poussée des technologies émergentes avec une simplification des interfaces utilisateur et une personnalisation approfondie, destinées à répondre de manière précise aux exigences spécifiques de chaque entreprise.

4. Méthodologie et Organisation

En ce qui concerne la méthodologie de développement, j'ai utilisé le concept agile en m'inspirant de la méthode SCRUM. J'ai structuré le travail en sprints de deux semaines, chaque sprint est dédié à l'accomplissement de tâches spécifiques. Cette approche m'a permis de diviser le projet en étapes gérables et itératives, facilitant ma gestion du temps disponible. Ce qui permet de fixer des objectifs pour chaque sprint et maintenir un rythme de progression.
Pour organiser mes tâches et suivre le déroulement des sprints, j'ai opté pour l'utilisation de l'outil Trello. Trello est une plateforme qui permet de créer des tableaux pour chaque sprint et d'y répertorier les tâches à accomplir, en cours de réalisation, et terminées. Grâce à son système de cartes, je peux attribuer des priorités à chaque élément. Cette utilisation de Trello permet une meilleure gestion du temps pour le projet.

Pendant les quatre mois de janvier à avril, j'ai défini des objectifs pour chaque mois sur la réalisation du projet. Étant seul sur le sujet, j'ai fait le choix de ne pas établir de Gantt, optant plutôt pour une approche flexible basée sur les 70 heures prévues. Bien que cette estimation puisse être dépassée compte tenu de l'étendue des différentes tâches à accomplir si je choisis d'aller au bout de mon idée. Voici la répartition mensuelle des objectifs fixés :

- Janvier : Durant ce mois, les objectifs étaient la rédaction des bases du rapport, incluant l'introduction, la partie motivation et étude du marché, ainsi que le détail du cahier des charges. Parallèlement, j'ai consacré du temps à la réflexion des fonctionnalités et des besoins ainsi que la création des maquettes de l'application pour visualiser son interface et son fonctionnement préliminaires.

- Février : Ce mois a été consacré au développement initial de l'application et à la mise en place de l'environnement technique nécessaire à son fonctionnement.Concrétiser les concepts définis dans le cahier des charges, en me concentrant sur la réalisation pratique de la solution mobile.

- Mars : Pendant ce mois, l'objectif principal était de continuer le développement de l'application en accordant une attention particulière à l'optimisation des fonctionnalités, à la résolution des problèmes techniques rencontrés et de s'assurer que la solution répond aux besoins spécifiés dans le cahier des charges.

- Avril : Le dernier mois du projet avait pour but de travailler sur les tests, la validation et la vérification du projet, ainsi qu'à la rédaction de la conclusion du rapport pour veiller à assurer la qualité et la fiabilité de la solution, tout en documentant les résultats et les conclusions dans le rapport final.

5. Cahier des charges

5.1. Analyse des besoins client

Les clients potentiels de la solution logicielle de répertoriage intelligent des projets sont les entités et organisations qui cherchent à améliorer leur gestion de l'information projet. Il peut s'agir de grandes entreprises, de PME, ou même d'organisations gouvernementales. Les clients potentiels peuvent être de divers secteurs comme l'industrie, les services, la santé, ou d'autres domaines qui ont besoin d'une gestion structurée de leurs projets.

Pour les organismes, différents types de personnes risquent d'utiliser la solution. Pour répondre aux attentes des utilisateurs finaux, il faut identifier les personas cibles qui représentent les profils types d'utilisateur de la solution.

Chef / Responsable :

- ➢ Utilisation : Gestion globale des projets
- ➢ Besoin :
  - Accès complet à tous les projets
  - Permission de créer, modifier et supprimer des projets.
  - Permission de visualiser et modifier la documentation/ressources des projets.
  - Accès aux rapports et aux données analytiques des projets.

Membre de l'entreprise (commerciaux, développeurs, …) :

- ➢ Utilisation : Visualisation des informations projets
- ➢ Besoin :
  - Accès aux projets de l'entreprise.
  - Accès aux documents et aux ressources liés aux projets.

Utilisateur Collaboratif (client, externe, …) :

- ➢ Utilisation : Visualisation des informations projets
- ➢ Besoin :
  - Accès aux projets auxquels il participe
  - Accès aux documents et aux ressources liés aux projets auxquels il participe.

La solution logicielle doit répondre aux besoins spécifiques de chaque persona, garantissant une expérience utilisateur optimale et une utilisation efficace de la solution.

5.2. Technologie utilisées

La solution utilisera un système de gestion de base de données (SGBD) afin de structurer l'ensemble des informations de manière efficace et pérenne. Le choix d'un SGBD robuste et évolutif est important pour garantir des performances optimales en termes de stockage, de récupération, et de gestion des données. Les critères de sécurité, la capacité à gérer des volumes importants de données, ainsi que la prise en charge de requêtes complexes sont des éléments à prendre en compte. C'est pourquoi Supabase sera utilisé comme solution SGBD principale. Cette décision découle de plusieurs facteurs, dont la nature open-source

de Supabase qui favorise la flexibilité et l'adaptabilité aux besoins spécifiques du projet. De plus, Supabase offre une architecture moderne et évolutive, permettant une gestion efficace des données tout en garantissant un niveau élevé de sécurité et de fiabilité.

En complément du SGBD, il y aura une base de données NoSQL Firebase. Cette plateforme sera pour la gestion des images, des flux d'informations à analyser, ainsi que des données statistiques. Cette approche hybride permettra de tirer parti des avantages spécifiques de chaque technologie, en fonction des besoins et des caractéristiques propres à chaque type de données.

L'intégration d'une API constitue un élément important de l'architecture du projet, facilitant la communication entre les différentes parties (BDD et applications) de notre système et permettant l'interopérabilité avec d'autres applications et services externes. L'utilisation d'une API offre une flexibilité accrue dans la gestion des données, des requêtes et des réponses, tout en favorisant la scalabilité et la modularité de notre solution. Pour la réalisation de cette API, la technologie Nest.js sera utilisée. Cette décision vient des avantages que donne Nest.js, comme sa structure modulaire, sa capacité à gérer efficacement les requêtes HTTP et son implémentation basée sur TypeScript qui favorise la maintenabilité et la lisibilité du code. Nest.js donne une intégration transparente avec d'autres bibliothèques et outils, facilitant ainsi le déploiement et la gestion de l'API dans un environnement de production et de test.

La plateforme de développement sélectionnée doit permettre une compatibilité avec les principaux langages de programmation et permettre une évolution future flexible de la solution. Elle doit être aussi compatible à la réalisation d'une solution mobile. Dans cette optique, le framework Flutter correspond le plus. Il est accompagné du langage Dart, qui est adapté à la réalisation d'une application mobile. Flutter offre une approche de développement cross-platform, ce qui signifie qu'il peut cibler simultanément les plateformes iOS et Android, réduisant ainsi les coûts de développement et simplifiant la gestion de code. De plus, la performance native de Flutter assure une expérience utilisateur fluide et réactive, tout en permettant une personnalisation avancée de l'interface utilisateur.

5.3. Fonctionnalités clés

La section suivante liste les principales fonctionnalités de l'application, conçues pour offrir une expérience utilisateur optimale et répondre aux besoins opérationnels d'une application de répertoriage projet.

- Un système de création et d'authentification sécurisé de compte utilisateur avec une fonctionnalité de mot de passe oublié.
- Les utilisateurs peuvent créer leur profil personnel et facilement modifier leurs informations, telles que leur nom, leur adresse e-mai, et mettre à jour leur photo de profil.
- Les utilisateurs peuvent ajouter de nouvelles entreprises, mettre à jour les détails existants, visualiser les informations détaillées et supprimer les entrées obsolètes.

- Les utilisateurs peuvent gérer efficacement tous les aspects de leurs projets. Ils peuvent créer de nouveaux projets, mettre à jour les détails existants, visualiser les informations détaillées de chaque projet, ainsi que supprimer les projets terminés ou obsolètes.
- Une page dédiée est mise en place pour permettre aux utilisateurs de visualiser tous les changements qui ont eu lieu dans les projets. Cette page informative offre une vue globale et chronologique des modifications de projet, les ajouts ou suppressions.
- L'ergonomie de l'interface utilisateur doit être conçue de manière à faciliter la navigation, minimiser le temps d'apprentissage, et permettre une utilisation efficace même pour les utilisateurs novices.
- Les éléments de l'interface, tels que les menus, les boutons, et les champs de saisie, doivent être uniformes et positionnés de manière logique pour une utilisation fluide
- La solution doit assurer le stockage sécurisé des données, en respectant des normes de confidentialité reconnues telles que le Règlement Général sur la Protection des Données (RGPD).

5.4. Contraintes et limites

La solution devra respecter des critères de performance pour garantir une expérience utilisateur optimale. Les temps de réponse des fonctionnalités clés, comme la recherche et la récupération des informations projets, ne devront pas excéder des seuils prédéfinis.
L'application doit pouvoir fonctionner sur tout type d'appareil, sous n'importe quelle OS (IOS/Android)

Malgré ses fonctionnalités avancées, la solution peut présenter des limites fonctionnelles. En cas de non-utilisation ou d'utilisation incorrecte, certains scénarios peuvent être rencontrés. Ces scénarios incluent, mais ne se limitent pas à, la perte de connexion réseau, des erreurs de saisie de données, ou des manipulations incorrectes par les utilisateurs. Des messages d'erreur informatifs seront intégrés pour guider les utilisateurs dans de ce genre de situations. Il est important que les utilisateurs soient informés des scénarios de non-utilisation potentiels pour minimiser les risques d'erreurs.

La solution sera conçue de manière à être adaptable aux évolutions futures des technologies, des besoins de l'entreprise, et des pratiques de gestion de projets. Mais, il est possible que des limites fonctionnelles surgissent en cas de changements majeurs dans l'infrastructure technologique, les standards de sécurité, ou les exigences opérationnelles. La documentation détaillée sur l'architecture et les dépendances sera fournie pour faciliter toute modification future.

6. Conception

6.1. Schéma global de l'architecture

L'architecture générale est une approche hybride, combinant deux bases de données distinctes (Sql avec Supabase et NoSQL avec supabase) qui sont pré-hébergées, une API intermédiaire qui est sur un serveur à part et une application mobile. Les données structurées sont stockées dans une base de données relationnelle, tandis que les données non structurées sont gérées via une base de données NoSQL. L'API agit comme un pont entre les bases de données et l'application mobile, facilitant la communication et permettant une manipulation sécurisée des données. Cette architecture donne une flexibilité et une extensibilité importante, permettant à la solution de s'adapter aux besoins changeants des utilisateurs tout en garantissant des performances optimales et une expérience utilisateur fluide. Pour plus de visibilité : **(*c.f annexe 4*)**.

Pour garantir une structure claire et cohérente de la base de données relationnelle, un diagramme détaillé décrivant les entités a été réalisé. Ce diagramme donne une vue d'ensemble sur la gestion de la base de données SQL. **(*c.f annexe 5*).**

Dans l'optique d'avoir une identité unique pour mon application, j'ai créé une charte graphique qui contient le logo sous différente forme (seul, avec titre, en mode sombre). Il contient les 3 couleurs principales et la typographie qui sera utilisée dans le projet : **(*c.f annexe 6*).**

6.2. User stories

Pour concevoir les maquettes de l'application, j'ai opté pour l'utilisation de l'outil Whimsical. Cette plateforme donne une interface intuitive, permettant de créer facilement des interfaces utilisateur. Grâce à Whimsical, j'ai pu visualiser de manière efficace les différentes fonctionnalités et interactions de mon application avant de passer à la phase de développement. Cela m'a permis d'avoir une meilleure compréhension de la manière dont les utilisateurs interagissent avec l'application et d'anticiper les besoins en termes de conception et de fonctionnalités.

Voici quelques-unes des principales histoires utilisateur (user stories) que l'on retrouvera dans l'application, chacune étant illustrée par des maquettes :

Inscription/Connexion **(*c.f annexe 1*)** :

- En tant qu'utilisateur, je veux pouvoir m'inscrire sur la plateforme en fournissant mon nom, mon adresse e-mail et un mot de passe sécurisé.
- En tant qu'utilisateur inscrit, je veux pouvoir me connecter à mon compte en utilisant mes identifiants (adresse e-mail et mot de passe).
- En tant qu'utilisateur, je veux recevoir un message d'erreur clair en cas de saisie incorrecte de mes informations d'identification lors de la connexion.
-

Création d'une Entreprise/Visualisation **(*c.f annexe 3*)** :

- En tant qu'utilisateur, je veux pouvoir créer une nouvelle entreprise en fournissant des informations telles que le nom, l'adresse et la description.
- En tant qu'utilisateur, je veux pouvoir visualiser l'entreprise auquelle je suis associé en tant qu'employé ou administrateur.
- En tant qu'utilisateur, je veux pouvoir accéder à la page d'une entreprise pour voir ses informations et ses membres.

Création d'un Projet/Visualisation **(*c.f annexe 2*)** :

- En tant qu'utilisateur connecté, je veux pouvoir créer un nouveau projet en fournissant un titre, une description et d'autres détails pertinents.
- En tant qu'utilisateur, je veux pouvoir visualiser tous les projets auxquels je participe ou que j'ai créés dans une liste claire et organisée.
- En tant qu'utilisateur, je veux pouvoir accéder à un projet spécifique pour voir ses détails et ses composants.

7. Réalisation / Mise en œuvre

7.1. Environnement de développement

Pour la réalisation de ce projet,j'ai choisi l'IDE IntelliJ pour plusieurs raisons. IntelliJ offre un environnement de développement robuste et hautement personnalisable, qui donne un ensemble complet d'outils pour la programmation en Flutter et en Nest.js. Grâce à ses fonctionnalités avancées telles que l'analyse statique du code, la complétion intelligente et le débogage intégré, IntelliJ a facilité le processus de développement en assurant une une qualité de code. En ce qui concerne la gestion des versions, j'ai opté pour GitHub Desktop. C'est une plateforme qui simplifie la gestion des branches, la fusion des modifications et la gestion des conflits.

Pour assurer une disponibilité de l'API auprès des utilisateurs finaux, j'ai choisi AWS EC2 pour héberger le serveur en mode production. AWS EC2 est une infrastructure cloud hautement évolutive et fiable qui garantit des performances optimales même dans des conditions de charge élevée. Avec les fonctionnalités de mise en réseau avancées d'EC2, j'ai pu configurer et déployer rapidement l'API, assurant une disponibilité continue et une réponse rapide aux requêtes des utilisateurs depuis l'application. En ce qui concerne la distribution de l'application mobile, j'ai opté pour l'utilisation d'APK pour partager la version finale de l'application avec les testeurs et les utilisateurs finaux. Les APK fournissent un moyen efficace de distribuer des applications Android en dehors du Google Play Store, permettant une diffusion rapide de l'application sur la plupart des gamme d'appareils Android. Cette approche a permis de faciliter le processus de test et de recueillir rapidement des retours d'utilisateurs, contribuant ainsi à améliorer la qualité et la convivialité de l'application.

7.2. Développement des fonctionnalités :

La conception de la structure de l'application a été soigneusement élaborée en suivant des principes de design patterns en Flutter, avec notamment l'utilisation de l'héritage pour la gestion des différents types de projets. Cette approche a permis d'établir une organisation claire et modulaire du code, donnant une base solide pour la maintenance future et l'ajout de nouvelles fonctionnalités. En structurant le code de cette manière, chaque composant de l'application est encapsulé dans des classes spécialisées, ce qui facilite sa réutilisation et sa compréhension, tout en favorisant une évolutivité optimale du projet. En plus de suivre des principes de design patterns en Flutter, j'ai également créé un ensemble de composants personnalisés que j'ai intégré dans la structure de l'application. Ces composants, développés, sont conçus pour être réutilisables et cohérents à travers toute l'application. Par exemple, j'ai mis en place un dossier dédié contenant des composants tels que des TextFields, des popups, des boutons, etc., chacun étant conçu avec un fonctionnement de base uniforme.

Cette méthode assure une expérience utilisateur homogène et familière à travers toutes les sections de l'application. En utilisant ces composants personnalisés, je garantie que les éléments d'interface utilisateur tels que les champs de texte, les fenêtres modales et les boutons ont une apparence et un comportement cohérents, ce qui améliore la convivialité de l'application et simplifie la maintenance du code. De plus, cela permet d'accélérer le processus de développement en réduisant la duplication du code et en favorisant la cohérence visuelle à travers l'ensemble de l'application.

En ce qui concerne l'architecture logicielle, j'ai opté pour le modèle MVC (Modèle-Vue-Contrôleur) en Flutter. Ce modèle a été choisi pour sa capacité à fournir une séparation claire des responsabilités entre les différentes parties de l'application. Le modèle représente les données et la logique métier, la vue gère l'interface utilisateur et le contrôleur coordonne les interactions entre les deux. Cette approche a permis de garantir une structure robuste et évolutive de l'application, facilitant ainsi la maintenance et l'extension du code au fil du temps.

Pour l'API, j'ai utilisé le framework Nest.js, en organisant le code selon un modèle par objet métier. Ça consiste à encapsuler chaque entité principale de l'application, telle que les utilisateurs, les projets et les entreprises, dans des modules dédiés. Chaque module est responsable de la gestion et de la manipulation des données liées à son objet métier spécifique. Cette organisation modulaire du code a permis une meilleure séparation des préoccupations et une gestion plus efficace des fonctionnalités, favorisant la maintenabilité de l'API dans le temps.

7.3.Intégration des technologies :

L'intégration des deux bases de données, Supabase et Firebase, a été une mise en place importante du développement, réalisé en suivant une approche API-first. Cette méthodologie a permis de créer une architecture solide où l'API joue un rôle central en servant de pont entre les deux bases de données. Cette conception assure une communication sécurisée entre l'application et les bases de données, garantissant une gestion efficace des données. Dans ce système, l'application communique exclusivement avec l'API pour accéder à l'ensemble de ses données. Cette approche permet une gestion centralisée des informations, simplifiant la logique d'accès aux données à travers l'application.

Lors de l'intégration, j'ai rencontré des défis liés à la configuration de Firebase sur l'API, principalement en raison de conflits de configuration avec Supabase. Ces conflits ont entraîné des difficultés dans la mise en place et la synchronisation des deux bases de données. Néanmoins, grâce à des ajustements techniques, j'ai pu surmonter ces problèmes et garantir un bon fonctionnement de l'ensemble du système. Ces ajustements ont impliqué une révision approfondie de la configuration et des paramètres de chaque base de données, ainsi que des modifications spécifiques pour résoudre les conflits rencontrés. En fin de compte, cette expérience a renforcé ma compréhension des deux plateformes et m'a permis d'optimiser l'intégration pour assurer la stabilité de l'application.

8. Tests / Validation / Vérification

8.1. Stratégie de tests

En ce qui concerne les tests sur Flutter, sur la partie application mobile, j'ai été confronté à un défi quant à la manière de tester l'application. Cependant, j'ai découvert que Flutter propose un module de test intégré, ce qui m'a permis de mettre en place une stratégie efficace. J'ai utilisé Flutter Test pour contrôler et vérifier le bon fonctionnement de mes formulaires ainsi que l'exactitude des données traitées par l'application. Cette approche m'a offert une manière structurée de tester les différentes fonctionnalités de l'application mobile.

Pour ce qui est de la partie API, étant donné qu'il s'agit essentiellement d'une API Gateway qui agit comme un simple relais de données, les tests unitaires classiques n'est pas le plus approprié. J'ai donc concentré les tests qui vérifient le bon fonctionnement des points d'entrée et de sortie de l'API. Cela m'a permis de m'assurer que les données sont correctement transmises et reçues par l'API.

En ce qui concerne la vérification du code et le maintien d'un code propre, j'ai utilisé les bibliothèques ESLint et Prettier sur l'API. Ces outils ont permis de garantir que le code respecte les normes de codage définies et qu'il reste cohérent et lisible.

8.2. Types de tests

Pour les tests sur Flutter :

- Des tests natifs ont été réalisés sur les formulaires de l'application, pour vérifier leur bon fonctionnement.
- J'ai également effectué des tests de monkey testing avec la participation de 9 utilisateurs, ce qui m'a permis d'identifier les éventuels problèmes d'utilisation et d'ergonomie de l'application.

Concernant les tests sur l'API :

- Des tests unitaires ont été mis en place avec Jest pour évaluer le bon fonctionnement des Endpoint de l'API.
- Des tests de performance ont été effectués avec Artillery afin de mesurer les performances de l'API sous différentes charges de travail. Artillery a fourni des statistiques détaillées sur les performances de l'API, telles que le temps de réponse moyen et les taux d'erreur.
- Des tests end-to-end ont été réalisés avec Postman pour vérifier le comportement complet de l'API, de l'entrée des données à la sortie.

8.3. Documentation du code

La documentation du code est une partie importante pour assurer la compréhension et la maintenabilité du projet. Sur l'application et l'API, j'ai veillé à ce que 90 % du code au minimum soit documenté. J'ai suivi les normes de documentation spécifiques définies pour le projet, garantissant ainsi que chaque fonction, classe et méthode soit correctement documentée.

9. Conclusion

Le projet présenté consiste en la conception et le développement d'une solution logicielle de répertoriage intelligent des projets, destinée à améliorer la gestion de l'information projet pour diverses organisations. Ce projet s'est déroulé sur une période définie, avec des objectifs précis à atteindre dans le cadre d'une approche agile inspirée de la méthode SCRUM. Durant cette période, j'ai mis en place une méthodologie rigoureuse de développement, en structurant le travail en sprints de deux semaines et en utilisant l'outil Trello pour la gestion des tâches. J'ai défini des objectifs mensuels, couvrant la rédaction du rapport, le développement de l'application, l'optimisation des fonctionnalités, et les tests de validation. J'ai également sélectionné les technologies appropriées pour chaque aspect du projet, en privilégiant des solutions robustes et évolutives telles que Supabase, Firebase, Nest.js et Flutter.

Malgré les progrès réalisés, certaines limitations ont été rencontrées au cours de cette étude. Le temps imparti n'a pas permis d'explorer pleinement les possibilités en termes d'UX/UI, de développement de fonctionnalités supplémentaires telles qu'un guide dans l'application ou un système de mise en maintenance et de réaliser des tests plus approfondis. Ces contraintes ont nécessité des compromis dans la portée du projet. Je voudrais aussi souligner que malgré les limitations rencontrées, ce projet a été une occasion précieuse d'apprentissage. La résolution des problèmes techniques, la prise de décision stratégique et la gestion des priorités ont été des compétences clés acquises tout au long du processus. De plus, la collaboration avec différentes technologies m'a permis d'approfondir mes connaissances et d'explorer de nouvelles possibilités dans le domaine du développement logiciel.

Si le projet est amené à aller plus loin, plusieurs perspectives d'évolution sont envisagées. Tout d'abord, je souhaite mettre l'application sur les stores mobiles pour la rendre accessible à un plus large public. De plus, si je devais recommencer ce projet, je mettrais en avant une meilleure gestion du temps et des tâches, en allant davantage à l'essentiel dès le départ. Enfin, je retiens l'importance du recul sur un projet individuel, qui permet d'identifier les points forts et les axes d'amélioration pour progresser efficacement.

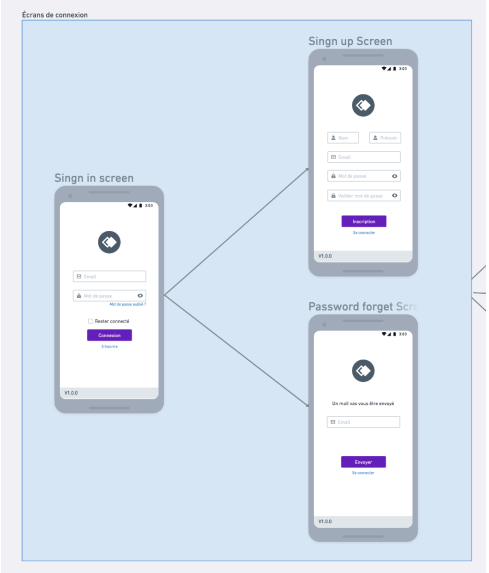
En conclusion, ce projet a été une expérience enrichissante qui a permis d'acquérir de nouvelles compétences, de surmonter des défis et d'identifier des opportunités de croissance. Avec une approche réfléchie, je suis convaincu que ce projet peut continuer à évoluer et à apporter de la valeur à beaucoup d’entreprises et utilisateurs.

8. Bibliographie

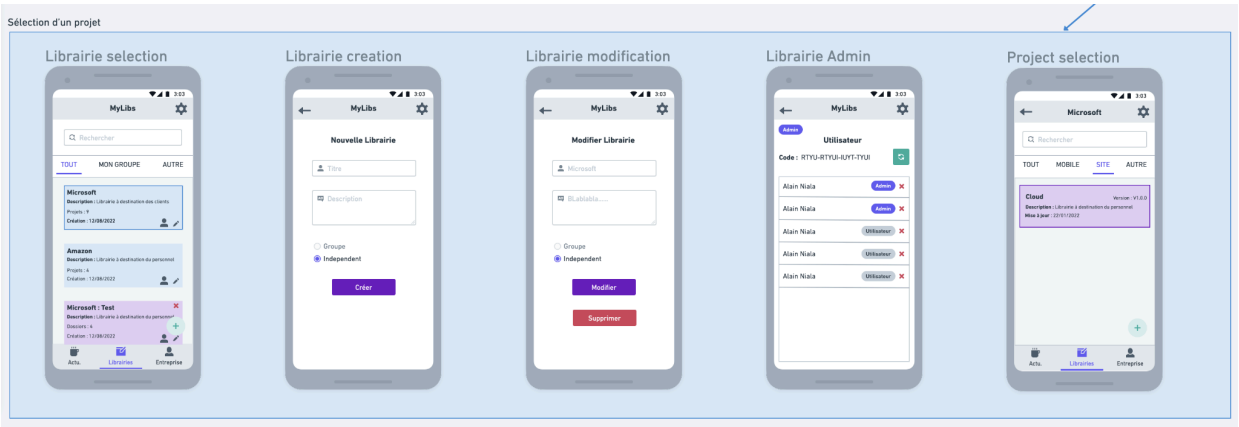
- CMMI (Capability Maturity Model Integration) : https://fr.wikipedia.org/wiki/Capability_Maturity_Model_Integration
- Théorie sur la gestion d'information : https://journals.openedition.org/questionsdecommunication/10030
- Microsoft Project Server : https://www.theprojectgroup.com/fr/outils-gestion-projet/microsoft-project-server
- Primavera P6 : https://www.oracle.com/fr/construction-engineering/primavera-p6/#rc30p1
- Assembla : https://get.assembla.com/
- Redmine : https://www.blogdumoderateur.com/tools/redmine/
- Supabase : https://supabase.com/
- Firebase : https://firebase.google.com/
- Flutter : https://flutter.dev/
- Nest.JS : https://nestjs.com/
- Trello : https://trello.com/fr
- whimsical : https://whimsical.com/

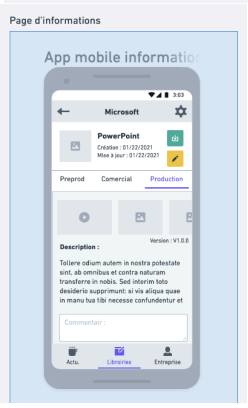
9. Annexes

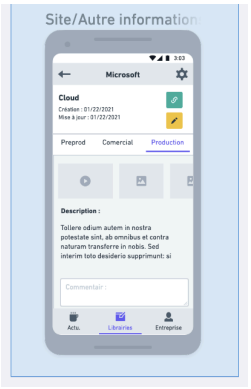
Annexe 1 : Maquette écrans inscription/connexion



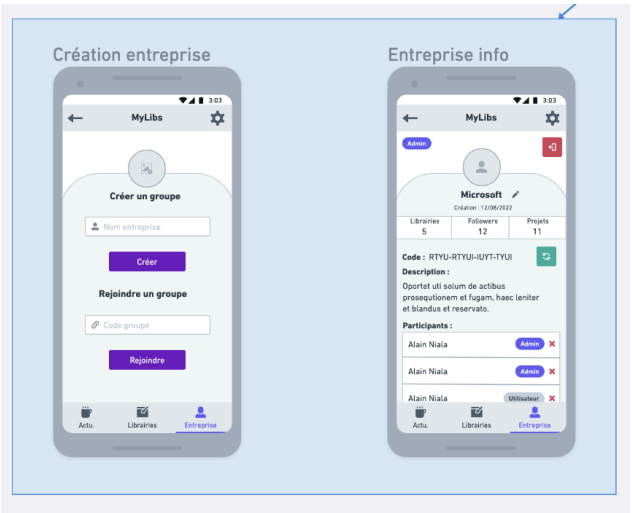
Annexe 2 : Maquette écrans création et visualisation des projets



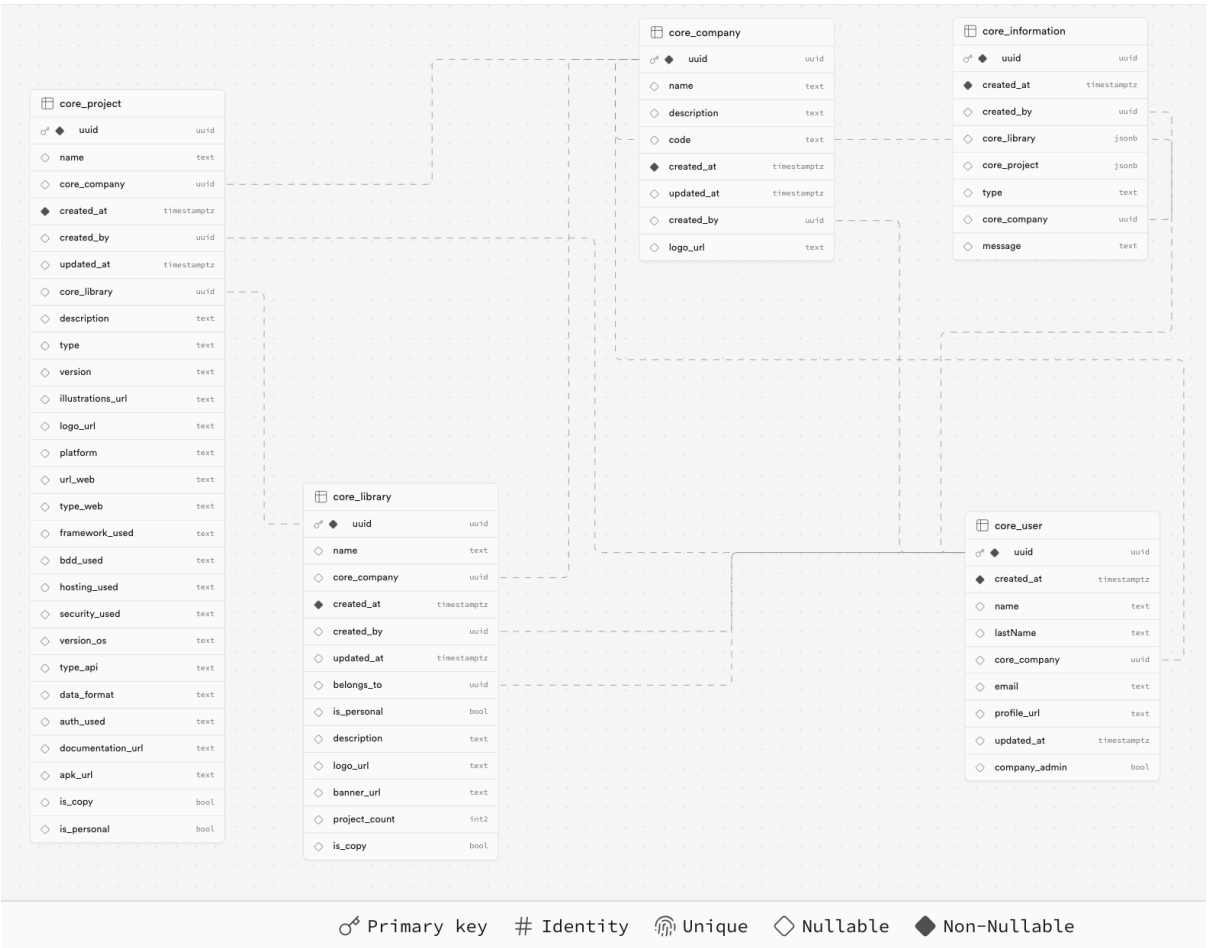




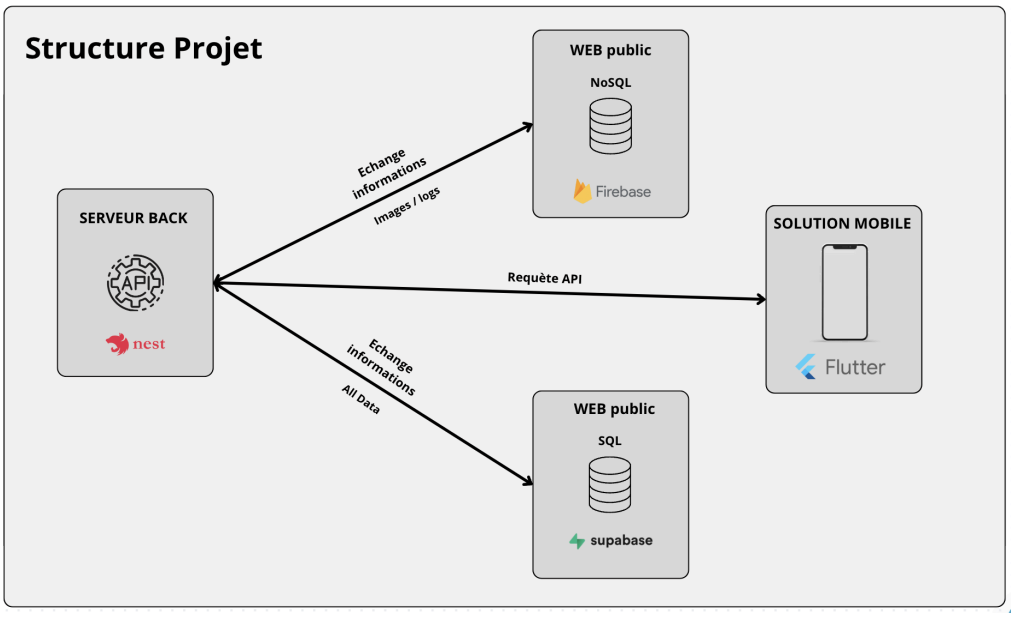
Annexe 3 : Maquette écrans gestion entreprise



Annexe 4 : Diagramme BDD sql (Supabase)



Annexe 5 : Structure du projet



Annexe 6 : Charte graphique