Introduction générale

Dans un monde en constante évolution, les avancées technologiques bouleversent notre quotidien et transforment les pratiques des entreprises. La révolution numérique a introduit de nouvelles manières de communiquer, de travailler et de faire des affaires. Les interfaces de programmation d'applications (API) sont devenues des éléments essentiels de ce paysage technologique, facilitant l'intégration et l'interopérabilité entre divers systèmes et applications. Elles permettent aux entreprises de rester compétitives en offrant des services plus efficaces et réactifs.

Vermeg, fondée en 2002, est une entreprise spécialisée dans les solutions logicielles pour les institutions financières, avec une présence dans dix-huit pays sur trois continents. Face aux défis posés par la complexité croissante des services financiers et la demande pour des solutions numériques performantes, Vermeg a entrepris de développer une marketplace d'API. Ce projet vise à centraliser et optimiser l'accès aux API de l'entreprise, améliorant ainsi l'expérience utilisateur et répondant aux besoins spécifiques de ses clients.

Le principal objectif de ce projet est de concevoir et de développer une marketplace d'API offrant une interface intuitive, des fonctionnalités avancées de gestion des abonnements, un suivi détaillé des statistiques d'utilisation et des mécanismes de sécurité robustes. En outre, la plateforme doit permettre la monétisation des API, ouvrant de nouvelles opportunités commerciales pour Vermeg.

Le développement d'une marketplace d'API représente une avancée stratégique pour Vermeg. En centralisant les API, l'entreprise peut améliorer l'efficacité de ses opérations internes, offrir une meilleure satisfaction client et explorer de nouvelles opportunités de croissance. Ce projet s'inscrit dans la vision de Vermeg de rester à la pointe de l'innovation technologique et de fournir des solutions de haute qualité adaptées aux besoins des institutions financières. Ce projet de fin d'études, réalisé par Yassine Lassoued et Ayoub Shili sous la supervision de M. Akrama Anaya et Mme Amel Triki, vise à démontrer notre capacité à concevoir et développer des solutions technologiques complexes et innovantes. La marketplace d'API est conçue pour être une solution robuste, scalable et sécurisée, répondant aux attentes de Vermeg et de ses clients. En réussissant ce projet, nous espérons contribuer significativement à l'innovation technologique au sein de Vermeg et dans le secteur financier en général.

Ce rapport détaille les différentes étapes suivies pour la réalisation de ce projet, de l'étude des besoins à la planification, en passant par la conception et la mise en œuvre technique de la solution. Nous espérons que ce travail apportera une valeur ajoutée significative à Vermeg et ouvrira de nouvelles perspectives pour l'avenir.

Chapitre 1

Présentation du cadre du projet

Plan

Introduction

Ce chapitre a pour objectif de situer notre travail par rapport à son cadre général. Nous commençons, tout d'abord, par la présentation de l'organisme d'accueil, puis nous passons à des notions fondamentales relatives aux API . Ensuite, nous abordons l'étude de marché et concluons par l'analyse de l'existant ainsi que la méthodologie et le langage de modélisation adoptés.

I. Présentation de la société

1. Présentation générale

Fondée en 2002, Vermeg est une entreprise off-shore comptant plus de 1600 collaborateurs, elle est indépendante de la société mère BFI établie en 1994. Vermeg dispose d'une expertise en monétique et en édition de logiciels bancaires et financiers, et offre une large gamme de logiciels pour les différents tiers des institutions financières concernés par le traitement des titres.
Les produits que Vermeg offre, sont utilisés dans dix-huit différents pays à travers trois continents. la figure 1.1 représente la dispersion géographique de ses différents clients. [1]

Figure 1.1 : Les clients de Vermeg dans le monde

2. Départements de Vermeg

Les principaux départements de Vermeg sont :

- **Recherche et Développement :** Se concentre sur l'innovation et le développement de nouvelles technologies et solutions.
- **Technologie et Développement de Produits :** Responsable de la conception, du développement et de la maintenance des produits logiciels.
- **Ventes et Marketing :** Chargé de la promotion des produits et des services, de la gestion des relations clients et de l'acquisition de nouveaux clients.
- **Support Client et Services Professionnels :** Offre une assistance technique et des services de consultation aux clients.
- **Ressources Humaines :** Gère le recrutement, la formation, et le développement des employés ainsi que les relations de travail.
- **Finance et Comptabilité :** Gère les finances de l'entreprise, y compris la comptabilité, la gestion des budgets et les rapports financiers.
- **Gestion de Projet :** Assure la planification, l'exécution et la clôture des projets en respectant les délais et les budgets.
- **Qualité et Conformité :** Veille à ce que les produits et services répondent aux normes de qualité et aux réglementations en vigueur.

Ces départements collaborent pour fournir des solutions technologiques complètes et innovantes aux clients de Vermeg dans les secteurs financiers et d'assurance.
Notre stage s'est effectué au sein du département Recherche et Développement.

3. Services

L'activité de Vermeg s'articule principalement autour de quatre axes : [2]

- Assurance : couverture d'assurance individuelle ou de groupe pour l'épargne et la santé.
- Gestion de richesses et d'actifs : gestion des décisions financières dédiée aux besoins spécifiques de gestion du patrimoine.
- Marché financier et services de sécurité.
- Services financiers digitaux : numérisation des processus financiers et analyse des données sensibles.

Vermeg fournit à ses clients un ensemble de logiciels spécialisés dans le domaine financier tels que :

- MEGARA (Securities processing) : La suite MEGARA est une plateforme modulaire pour le traitement des titres destinée essentiellement aux institutions financières qui propose un nombre de modules pouvant être implémentés séparément.
- PALMYRA : est un framework JEE compatible SOA (architecture orientée service) qui incorpore des composants ainsi que des services web réutilisables. L'implémentation de services web réutilisables permet de développer des logiciels de meilleure qualité en temps réduit dont l'architecture est basée sur un client léger.
- SOLIFE : Une solution d'administration de polices d'assurance vie et un portail web à destination des clients finaux/brokers.
- SOLIAM : est une solution de gestion de portefeuilles pour les gestionnaires d'actifs institutionnels de fortune. Ayant une présence internationale en Belgique, France, Irlande, Luxembourg, Pays-Bas, Suisse, Royaume-Uni, Tunisie et des clients dans plus de 23 pays, VermegLife et Soliam sont considérés comme les deux produits phares de Vermeg BSB.

Maintenant que nous avons exploré la structure, les départements et les services de l'entreprise d'acceuil, et étant donné que Vermeg génère de nombreuses API et que notre projet s'inscrit dans ce contexte, nous allons aborder les concepts clés liés aux API.

II. Concepts clés autour des API

Les API sont devenues des éléments essentiels du développement web moderne, permettant aux applications de communiquer et de partager des données. C'est pourquoi cette partie s'attache à présenter les fondements des API, leur documentation ainsi que le concept d'une marketplace d'API.

1. Les Fondements des API

1.1. Définition d'une API

Une API (Application Programming Interface) permet de rendre disponibles les données ou les fonctionnalités d'une application existante afin que d'autres applications les utilisent. Elle facilite le partage et l'intégration de fonctionnalités dans des architectures existantes. En pratique, une API agit comme un pont d'accès vers une fonction spécifique gérée par une entité distincte. [3]

1.2. Fonctionnement d'une API

Une API fonctionne généralement selon le principe requête-réponse :

- Requête : Le client envoie une requête à l'API, en précisant l'action souhaitée et les données nécessaires.
- Réponse : L'API traite la requête et envoie une réponse au client, contenant les données ou les résultats de l'action demandée.

1.3. Types d'API

Les interfaces de programmation d'applications ne sont pas toutes créées de la même manière. Chaque type d'API a une utilité spécifique et s'intègre de manière différente dans l'écosystème des applications et des services. Voici les principaux types d'API selon les styles d'architecture :

a. API REST (Representational State Transfer)

Une API Rest est caractérisée par :

- Architecture client-serveur sans état
- Accès aux ressources via des URL et des méthodes HTTP (GET, POST, PUT, DELETE)
- Formats de données courants : JSON, XML
- Facile à utiliser et à comprendre
- Flexible et évolutive [4]

b. API SOAP (Simple Object Access Protocol)

Une API SOAP est caractérisée par :

- Architecture orientée service basée sur XML
- Utilisation de messages structurés et de protocoles web
- Norme plus complexe et plus lourde que REST
- Adaptée aux applications d'entreprise nécessitant une sécurité et une fiabilité élevées [5]

c. API RPC (Remote Procedure Call)

Une API RPC est caractérisée par :

- Exécution des fonctions à distance sur un autre serveur
- Utilisation des protocoles spécifiques comme XML-RPC ou JSON-RPC
- Moins répandue que REST et SOAP, mais utile pour certaines applications telles que les systèmes distribués, les microservices, et les environnements nécessitant des appels de procédure rapides et directs [6]

d. API GraphQL

Une API graphQL est caractérisée par :

- ➢ Requêtes basées sur un schéma flexible
- ➢ Permet de récupérer des données précises et structurées
- ➢ Plus récent et plus moderne que REST et SOAP
- ➢ A Gagné en popularité pour son efficacité et sa flexibilité [7]

1.4. Structure d'une API

La structure d'une API comprend plusieurs éléments essentiels qui facilitent la communication entre le client et le serveur. Voici les principaux composants :

- ➢ **Endpoint :** C'est l'URL à laquelle l'API peut être accédée. Chaque endpoint correspond à une fonction spécifique de l'API. Par exemple, https ://api.example.com/users peut être un endpoint pour accéder aux utilisateurs.
- ➢ **Méthodes HTTP :** Les actions possibles sur les ressources sont définies par les méthodes HTTP. Les plus courantes sont : GET, POST, PUT, DELETE
- ➢ **En-têtes (Headers) :** Informations supplémentaires envoyées avec les requêtes et les réponses HTTP, souvent utilisées pour l'authentification, la gestion du cache, le contrôle des formats de données, etc.
- ➢ **Paramètres :**
  - ❍ **Path Parameters** Inclus dans l'URL pour spécifier une ressource particulière, par exemple, /users/id.
  - ❍ **Query Parameters :** Inclus dans l'URL après le symbole `?` pour filtrer ou modifier la requête, par exemple, `?sort=asc&limit=10`.
  - ❍ **Body Parameters** Paramètres de corps (Body Parameters) : Utilisés principalement avec POST et PUT pour envoyer des données au serveur dans le corps de la requête.
- ➢ **Réponses :** Responses : Les réponses d'une API incluent le code de statut HTTP (par exemple, 200 pour le succès, 404 pour non trouvé, 500 pour une erreur serveur) et le corps de la réponse contenant les données ou les messages d'erreur

Maintenant que nous avons examiné les fondements et les types d'API, il est essentiel de comprendre l'importance de la documentation qui est un élément crucial pour garantir la bonne utilisation et intégration d'une API.

2. Documentation d'une API

La documentation d'une API fournit aux développeurs et aux utilisateurs les clés pour exploiter pleinement le potentiel d'une API. En détaillant les endpoints, les méthodes, les paramètres, les réponses, les processus d'authentification, les exemples d'utilisation et la gestion des erreurs, elle éclaire les fonctionnalités de l'API et facilite son adoption.

2.1. Choix de la documentation

Dans le domaine de la documentation des API, plusieurs formats sont utilisés, tels que Swagger/ OpenAPI, RAML, API Blueprint . Cependant, Swagger/OpenAPI est, en effet, l'un des formats les plus courants et les plus populaires.

2.2. Swagger/OpenAPI

Swagger a débuté comme un ensemble d'outils open-source développés par SmartBear Software pour simplifier la conception et la documentation des API RESTful. Avec le temps, il s'est transformé en un ensemble de spécifications et de formats permettant de décrire les API de manière claire et cohérente. En 2015, SmartBear Software a cédé la spécification Swagger à la fondation OpenAPI Initiative, qui l'a renommée OpenAPI Specification (OAS). Cette transition a élevé OpenAPI au rang de norme de l'industrie pour la documentation des API, offrant un standard largement adopté pour décrire les API de manière précise et uniforme. [8]

2.3. Caractéristiques d'un fichier Swagger/OpenAPI

OpenAPI est une spécification de description d'API ouverte et standardisée. Elle permet de décrire les fonctionnalités d'une API de manière détaillée, notamment les endpoints, les paramètres, les réponses, etc...

OpenAPI fournit un format standardisé pour la documentation des API. Les développeurs utilisent des fichiers au format JSON ou YAML pour décrire les API de manière claire et compréhensible.[9]

La figure suivante présente un extrait d'un fichier swagger.

```json
{
  "openapi": "3.0.0",
  "info": {
    "title": "Exemple API",
    "description": "Cette API fournit des fonctionnalités pour créer un nouvel utilisateur.",
    "version": "1.0.0"
  },
  "servers": [
    {
      "url": "https://api.example.com/v1"
    }
  ],
  "paths": {
    "/users": {
      "post": {
        "summary": "Créer un nouvel utilisateur",
        "requestBody": {
          "required": true,
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/User"
              }
            }
          }
        },
        "responses": {
          "201": {
            "description": "Utilisateur créé avec succès."
          },
          "400": {
            "description": "Requête invalide. Veuillez vérifier les données envoyées."
          }
        }
      }
    }
  },
```

Figure 1.2 : Exemple de fichier swagger

Ce fichier Swagger décrit une API qui permet de créer un nouvel utilisateur via la méthode POST. Lorsqu'un utilisateur est créé avec succès, la réponse sera un code 201 avec un message indiquant le succès de l'opération. En cas de requête invalide, la réponse sera un code 400 avec un message informant l'utilisateur de vérifier les données envoyées.

2.4. Avantages de Swagger/OpenAPI

OpenAPI offre une documentation claire et cohérente pour les API, favorisant ainsi l'assimilation et l'adoption grâce à une documentation bien structurée. De plus, en adoptant OpenAPI, les organisations contribuent à la standardisation et à l'interopérabilité des API, simplifiant ainsi leur intégration avec différentes applications et systèmes.

Bien que d'autres standards de documentation d'API, tels que RAML et API Blueprint, existent, Swagger et OpenAPI sont généralement préférés dans l'industrie en raison de leur large adoption, de leur écosystème d'outils et d'intégrations riches, ainsi que de leur soutien actif de la communauté.[10]

2.5. Les outils de Swagger/OpenAPI

Plusieurs outils sont utilisés dans l'écosystème Swagger, tels que Swagger UI, Swagger Codegen et Swagger Editor. Swagger UI est utilisé pour afficher une interface interactive permettant aux développeurs et aux utilisateurs d'explorer facilement les endpoints, les paramètres et les réponses de l'API, améliorant ainsi la compréhension de son fonctionnement et favorisant son adoption.[11]

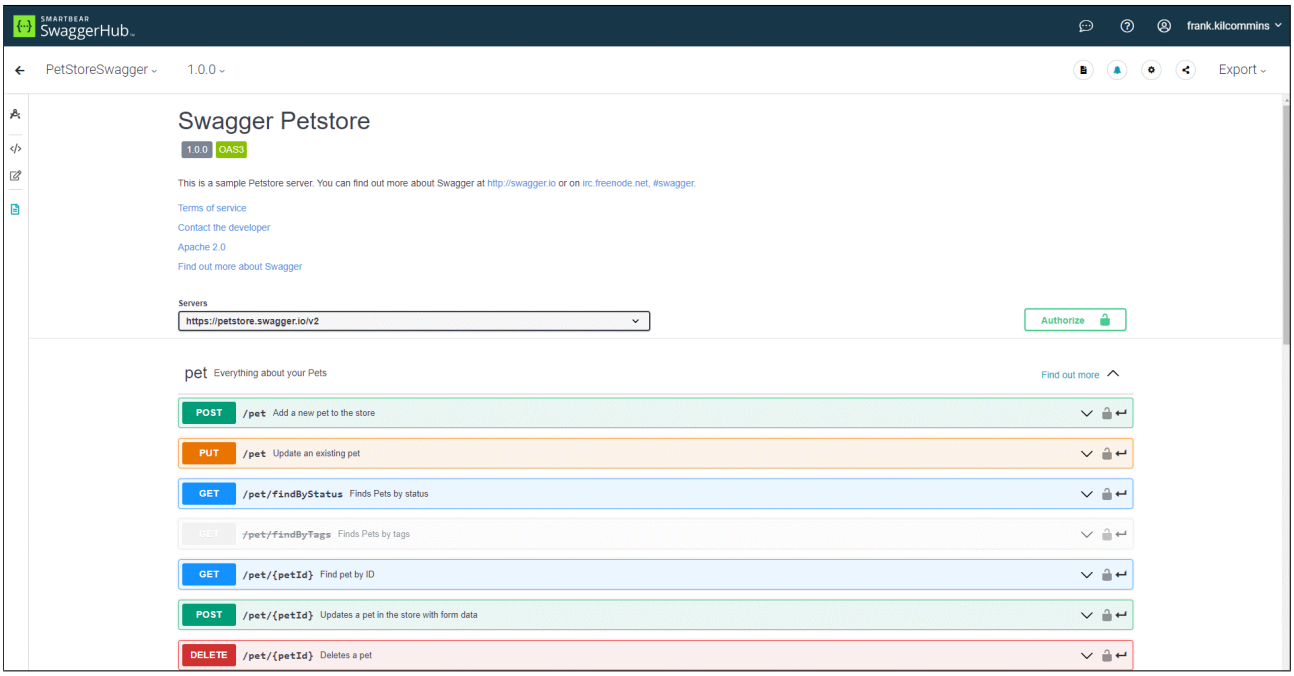

Figure 1.3 : Interface de Swagger UI

La documentation détaillée des API est cruciale pour leur adoption et leur utilisation efficace. Elle fournit une compréhension claire des fonctionnalités et des usages des API, facilitant leur intégration dans divers projets de développement. Cependant, la documentation seule ne suffit pas toujours à rendre les API largement accessibles. C'est ici qu'intervient le concept de marketplace d'API, qui offre un espace centralisé où les développeurs peuvent non seulement trouver et utiliser des API, mais aussi les commercialiser et les monétiser publiquement, rendant ainsi les API plus accessibles et favorisant une adoption plus large.

3. MarketPlace d'API

3.1. Définition d'une API MarketPlace

Une marketplace d'API est une plateforme en ligne dédiée aux développeurs et aux entreprises pour découvrir, acheter, vendre et intégrer des API.

Elle facilite l'échange de services et de données entre différents systèmes en offrant une plateforme centralisée pour les API, permettant ainsi aux fournisseurs de monétiser leurs API et aux consommateurs de trouver facilement les solutions dont ils ont besoin comme illustre la figure suivante :

Figure 1.4 : Processus de marketplace d'API

3.2. Fonctionnalités d'une API Marketplace

Les fonctionnalités d'une API Marketplace varient d'une plateforme à l'autre, mais les plus courantes incluent :

- **Recherche d'APIs :** Les développeurs d'applications peuvent rechercher des APIs par nom, par catégorie, par fonctionnalité, etc.
- **Consultation des détails des APIs :** Les développeurs d'applications peuvent consulter les détails d'une API, tels que sa documentation, ses exemples de code, ses tarifs, etc.
- **Abonnement aux APIs :** Les développeurs d'applications peuvent s'abonner aux APIs qu'ils souhaitent utiliser.
- **Consommation des APIs :** Les développeurs d'applications peuvent utiliser les APIs auxquelles ils sont abonnés pour ajouter des fonctionnalités à leurs applications.
- **Gestion des APIs :** Les fournisseurs d'API peuvent gérer leurs APIs sur la plateforme, en modifiant leurs informations, en ajoutant de nouvelles fonctionnalités et endpoints, etc.

3.3. Avantages d'une API marketplace

Les API Marketplaces offrent plusieurs avantages tant pour les fournisseurs d'API que pour les développeurs d'applications :

Pour les Fournisseurs d'API :

- ➢ Augmentation de la visibilité de leurs APIs, ce qui peut conduire à une plus grande adoption et utilisation.
- ➢ Accès à un large bassin de développeurs d'applications intéressés par l'intégration de nouvelles fonctionnalités.
- ➢ Possibilité de générer des revenus grâce à la vente des APIs proposées sur la plateforme.

Pour les consommateurs d'API :

- ➢ Facilité à trouver et à intégrer les APIs nécessaires à leurs projets, grâce à une offre diversifiée et bien référencée.
- ➢ Économie de temps et d'argent en utilisant des APIs déjà existantes, évitant ainsi le développement de fonctionnalités complexes.
- ➢ Ajout de nouvelles fonctionnalités à leurs applications sans nécessiter un développement supplémentaire, grâce à la disponibilité d'API prêtes à l'emploi.
- ➢ L'intégration de ces avantages dans une API Marketplace favorise la collaboration et l'innovation dans l'écosystème des développeurs d'applications.

4. Acteurs du domaine métier

Les principaux acteurs du domaine métier des marketplaces d'API sont les suivants :

- ➢ **Fournisseurs d'API :** Les fournisseurs d'API sont les entreprises ou les organisations qui développent et publient des APIs.
- ➢ **Consommateur d'API :** Les développeurs d'applications sont les utilisateurs qui utilisent les APIs pour ajouter des fonctionnalités à leurs applications.
- ➢ **Plateformes des marketplaces d'API :** Les plateformes des marketplaces d'API sont les plateformes en ligne qui mettent en relation les fournisseurs d'API et les développeurs d'applications.

III. Etude de marché

Cette étude constitue une partie importante de la phase d'analyse d'un projet qui pourra nous orienter pour tirer des avantages de ces solutions et remédier à leurs inconvénients. Cette étape est primordiale pour la mise en route de tout projet informatique ou autre. Parmi les marketplaces existantes, voici une comparaison de certaines d'entre elles : RapidAPI, API Layer, OpenAPIHub et Zyla Labs.

Tableau 1.1 : Etude comparative des marketplace d'API

Critère	**RapidAPI**	**API Layer**	**OpenApiHub**	**Zyla Labs**
Intermédiaire	Oui	Oui	Oui	Oui
Documentation sur les API	Oui	Oui	Oui	Oui
Tarification pour les consommateurs	Gratuit, Payant par nombre de requêtes pendant une durée précise	Gratuit, Payant par nombre de requêtes pendant un mois	Gratuit, Payant par nombre de requêtes pendant un mois	Gratuit, Payant par nombre de requêtes pendant un mois ou par an
Frais de la plateforme (Commission)	20%	15%	Gratuit (20%), Essential (12%), Professional (8%), Business, Enterprise	10%
Ajout d'endpoints	Manuellement	À travers la documentation Swagger	À travers la documentation Swagger	Manuellement
Test d'exemple sur la plateforme	Oui	Oui	Oui	Oui
Test de performance de l'API (latence)	Oui	Non	Oui	Oui
Communauté	Oui	Non	Oui	Oui
Signalement d'erreur	Non	Oui	Non	Non

Notification	Oui, mais pas en temps réel	Non	Non	Non
Mettre à niveau un plan	Ancient credit perdue	Ancient credit perdue	Ancient credit perdue	Montant déduit Selon Temps/Requêtes non Utilisés
Mode de paiement	Stripe	Stripe	Stripe	Stripe

Notes :

- **"intermédiaire" :** désigne ici le rôle de la plateforme en tant qu'entité permettant la transaction entre les fournisseurs et les utilisateurs d'API.
- **"Manuellement" :** signifie que les endpoints sont ajoutés manuellement, tandis que "Via Swagger" indique qu'ils sont ajoutés à travers la documentation Swagger.
- **"Tarification pour les consommateurs" :** Les utilisateurs ont la possibilité de choisir entre un modèle gratuit et un modèle payant basé sur le nombre de requêtes effectuées pendant une période définie.

Cette étude de marché comparative nous a permis de passer en revue quatre marketplaces d'API : RapidAPI, API Layer, OpenApiHub, et Zyla Labs. Chaque plateforme présente des avantages et des inconvénients spécifiques. RapidAPI, API Layer, OpenApiHub et Zyla Labs proposent des modèles de tarification similaires, mais elles ne sont pas flexibles. En ce qui concerne les commissions prélevées par la plateforme, celles-ci varient entre 20 % et 10 % .

Pour l'ajout d'endpoints, RapidAPI et Zyla Labs nécessitent un ajout manuel, alors que API Layer et OpenApiHub utilisent la documentation Swagger pour faciliter ce processus. Toutes les plateformes permettent de tester des exemples d'API, mais seuls RapidAPI, OpenApiHub, et Zyla Labs offrent des tests de performance de l'API en termes de latence.

Concernant le signalement d'erreur, seul API Layer offre cette fonctionnalité. RapidAPI est la seule plateforme à fournir des notifications, mais pas en temps réel. En termes de mise à niveau des plans, RapidAPI, API Layer et OpenApiHub annulent les crédits non utilisés lors de la mise à niveau, tandis que Zyla Labs déduit les montants selon le temps ou les requêtes non utilisés. Enfin, toutes les plateformes utilisent Stripe comme mode de paiement, garantissant ainsi une solution de paiement sécurisée et reconnue.

IV. Etude de l'existant

1. Déscription et critique de l'existant

Actuellement, au sein de Vermeg, le département recherche et développement travaille sur le développement d'API internes, notamment des modèles de machine learning, entre autres. Cependant, ces API restent en grande partie internes à l'entreprise et ne sont pas accessibles au public ou aux clients externes. Cette situation limite la capacité de Vermeg à tirer pleinement parti de son potentiel technologique et à générer des revenus supplémentaires en monétisant ces services. De plus, l'absence d'une plateforme de gestion et de distribution d'API adéquate entrave la mise à disposition efficace de ces services auprès du public ou des clients externes et de les monétiser.

2. Solution proposée

Nous proposons de créer une marketplace d'API, nommée InfinityAPI, qui permettra aux développeurs de publier leurs API sur la plateforme et de les monétiser. Cette marketplace offrira un espace centralisé où les développeurs pourront commercialiser leurs services auprès d'un large public, tout en offrant aux utilisateurs un accès simplifié à une gamme diversifiée d'API.
InfinityAPI est conçue en tenant compte des études de marché sur les marketplaces existantes. Parmi les avantages offerts par notre plateforme, nous proposons une intégration simplifiée des API grâce à l'utilisation de fichiers Swagger, une tarification flexible en termes de nom, de prix et de nombre de requêtes, des notifications en temps réel, et la possibilité d'upgrader les plans sans perte de crédits. De plus, notre plateforme impose une commission modérée compte tenu des fonctionnalités proposées.
En adoptant cette approche, Vermeg et d'autres entreprises pourront non seulement rentabiliser leurs investissements dans le développement d'API, mais aussi générer des revenus supplémentaires.

V. Méthodologie adoptée et langage de modélisation

Il existe plusieurs méthodologies de développement, chacune ayant ses propres avantages et inconvénients. Le choix de la bonne méthode dépend des exigences du projet, de la taille de l'équipe, des objectifs, des ressources disponibles et du calendrier.

1. Choix de la méthodologie

Pour notre projet de fin d'études "InfinityAPI", réalisé en binôme, il est crucial d'adopter une méthodologie de développement qui permette de s'adapter rapidement aux exigences du product owner, de livrer efficacement les fonctionnalités prioritaires et de favoriser une collaboration étroite

entre nous . Afin de répondre à ces besoins, nous avons choisi d'implémenter la méthodologie Agile, plus spécifiquement Scrum.

Scrum permet de développer notre solution de manière itérative et incrémentale, en se concentrant sur des cycles de développement courts appelés sprints. Cette approche garantit une livraison régulière de fonctionnalités à forte valeur ajoutée et offre la flexibilité nécessaire pour ajuster notre travail en fonction des changements de priorités et des besoins du product owner.

En adoptant Scrum, nous assurons une communication continue et une meilleure collaboration au sein de notre binôme, tout en maintenant une qualité élevée du produit final. Cette méthodologie nous permet de répondre efficacement aux exigences du marché, de livrer fréquemment des fonctionnalités et de garantir la satisfaction du product owner et des utilisateurs finaux. [12]

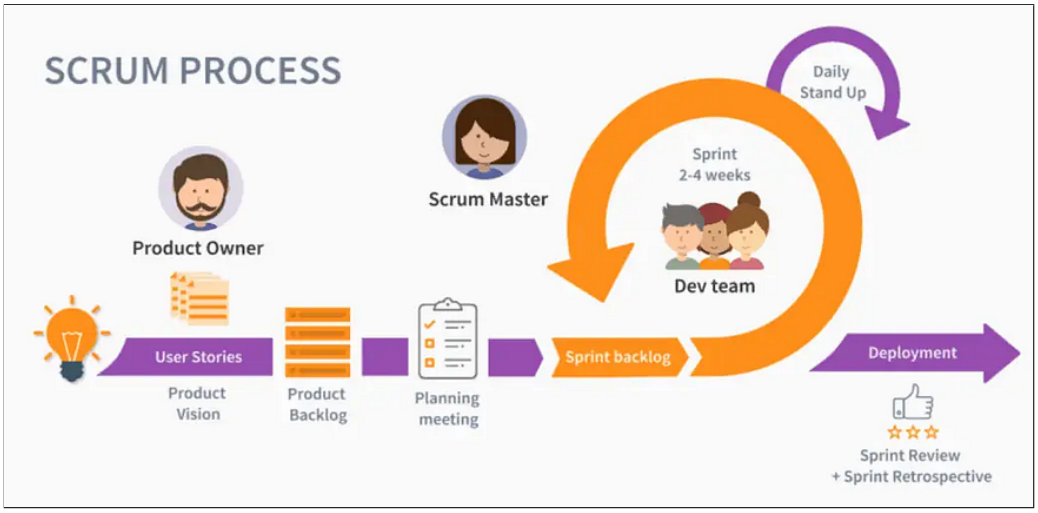

Figure 1.5 : Processus de Scrum

2. Language de modelisation

Le Unified Modeling Language (UML) a été conçu pour être un langage de modélisation visuel. Il est prévu pour l'architecture, la conception et l'implémentation de systèmes logiciels complexes par leur structure ainsi que leur comportement. UML dispose d'applications qui vont au-delà du développement logiciel, en particulier pour les flux de processus dans l'industrie. Il ressemble aux plans utilisés dans d'autres domaines et se compose de différents types de diagrammes. D'une façon générale, les diagrammes UML décrivent les frontières, la structure et le comportement du système et de ses objets. [13]

Conclusion

Dans cette section, nous avons introduit « Vermeg », notre organisme d'accueil. Ensuite, nous avons examiné les concepts clés liés aux API, réalisé une étude de marché, et enfin, discuté du choix des méthodes de développement et du langage de modélisation .Dans le deuxieme chapitre , nous aborderons la planification du projet.

Bibliographie

[1] Vermeg. « Présentation générale. » [Accès le 06-Mars-2024]. (Mars. 20224), adresse : https://www.vermeg.com/who-we-are/.

[2] Vermeg. « Présentation générale. » [Accès le 06-Mars-2024]. (Mars. 20224), adresse : https://www.vermeg.com.

[3] API. « Définition d'une API. » [Accès le 06-Mars-2024]. (Mars. 20224), adresse : https://www.salesforce.com/fr/resources/definition/api/.

[4] API. « Définition d'une API Rest. » [Accès le 06-Mars-2024]. (Mars. 20224), adresse : https://restfulapi.net/.

[5] API. « Définition d'une API SOAP. » [Accès le 06-Mars-2024]. (Mars. 20224), adresse : https://www.w3schools.com/xml/xml_soap.asp.

[6] API. « Définition d'une API RPC. » [Accès le 06-Mars-2024]. (Mars. 20224), adresse : https://www.jsonrpc.org/specification.

[7] API. « Définition d'une API Graphql. » [Accès le 06-Mars-2024]. (Mars. 20224), adresse : https://www.jsonrpc.org/specification.

[8] Swagger/OpenAPI. « Définition de swagger. » [Accès le 06-Mars-2024]. (Mars. 20224), adresse : https://swagger.io/about/.

[9] Swagger/OpenAPI. « Caractéristiques de swagger. » [Accès le 06-Mars-2024]. (Mars. 20224), adresse : https://swagger.io/docs/.

[10] Swagger. « Avantage d'un fichier Swagger/OpenAP. » [Accès le 06-Mars-2024]. (Mars. 20224), adresse : https://www.ionos.fr/digitalguide/sites-internet/developpement-web/quest-ce-que-openapi/.

[11] Swagger/OpenAPI. « Outil de swagger. » [Accès le 06-Mars-2024]. (Mars. 20224), adresse : https://swagger.io/tools/.

[12] Scrum. « Qu'est-ce que Scrum ? » [Accès le 06-Mars-2024]. (Mars. 20224), adresse : https://www.atlassian.com/fr/agile/scrum.

[13] UML. « Qu'est-ce que UML ? » [Accès le 06-Mars-2024]. (Mars. 20224), adresse : https://www.lucidchart.com/pages/fr/langage-uml.