



RAPPORT DE PROJET DE FIN D'ETUDES

En vue de l'obtention du

Diplôme National d'ingénieur Informatique

Option : Génie Logiciel-Architecture Logicielle

Département Informatique

Elaboré par

Ahmed BELAIBA

Au sein de

redlean

Création d'une solution intelligente pour l'automatisation de génération des dépenses

Soutenu le 03/10/2020 devant le jury :

Dr CHERIF Farouk

Dr BAHRIA Nihed

Dr KANZARI Dalel

M. BAHROUN Nader

Président

Examineur

Encadrante académique

Encadrant professionnel

Année Universitaire : 2019/2020

Code sujet : FI-GL20-075

Dédicace

A mon cher père qui a cru en moi et m'a toujours soutenu dans les périodes les plus difficiles de ma vie, pour son dévouement, son amour inconditionné et ses sacrifices pour faire de moi la personne que je suis aujourd'hui.

A ma maman qui m'a soutenu tout au long de mon parcours.

A Mme Noura et mon ami Haythem qui m'ont aidé et soutenu durant ce stage.

A toutes les personnes qui ont croisé mon chemin, ceux qui sont encore là et ceux que le temps a arraché de ma route.

A tous mes amis, pour leur confiance et leur soutien. Je tiens à leur dédier cet humble travail, fruit d'un long travail laborieux.

Remerciements

Je voudrais remercier tous ceux qui, sans leur aide inestimable, ce projet n'aurait jamais été mené à son terme. Mes remerciements s'adressent particulièrement à :

- Mon encadrent «**M. Nader BAHROUN**», Project Manager de REDLEAN services, à qui je tiens à exprimer toute ma gratitude pour l'aide qu'il m'a apporté durant toutes les phases du stage. Ses conseils étaient très précieux pour atteindre les objectifs de ce projet dans les délais convenus.
- Mon superviseur «**Mme Dalel KANZARI**» d'avoir accepté de diriger ce travail. Je la remercie pour son assistance, sa disponibilité, ses encouragements et son soutien. Ses conseils continus ont été déterminants dans l'élaboration de ce travail.
- «**Mme Noura ELABED** », Chef de projet Salesforce de Redlean services, à qui je tiens à exprimer toute ma gratitude pour la confiance qu'elle m'a accordé dès mon arrivée à l'entreprise.
- Tous les enseignants qui ont participé à mon évolution scientifique durant les années de formation à l'Institut supérieur des sciences appliquées et de technologie de Sousse «**(ISSATSO)**».
- A toutes les personnes qui ont aidé à l'élaboration de ce travail. Il n'est malheureusement pas possible de les citer tous ici. Je leur adresse mes plus profonds remerciements.
- Mes derniers mots s'adressent à tous les membres du jury pour l'honneur qu'ils me font en participant à l'examen de ce travail.

Tables des Matières

Introduction générale.....	1
Chapitre 1 : Contexte générale	3
1.1 Cadre du projet	3
1.2 Organisme d'accueil : Redlean services	3
1.2.1 Département d'accueil : Nouvelles technologiques.....	4
1.3 Etude de l'existant	5
1.3.1 Description de l'existant	5
1.3.2 Critique de l'existant.....	5
1.3.3 Solution proposée.....	5
1.4 Organisation.....	5
1.4.1 Méthode agile.....	5
1.4.2 Planification	6
Chapitre 2 : Etude préalable	6
2.1 Etude des concepts clés	7
2.2 Etude des solutions similaires.....	11
2.2.1 Expensya	11
2.2.2 N2F	11
2.2.3 Rydoo Expense	12
Chapitre 3 : Analyse et spécification des besoins	13
3.1 Analyse des besoins	13
3.1.1 Identification des acteurs	13
3.1.2 Besoins fonctionnels	13
3.1.3 Besoins non-fonctionnels.....	14
3.2 Analyse semi-formelle des besoins	14
3.2.1 Spécification des besoins	14

3.2.2 Cas d'utilisation «Traiter une note de frais»	15
3.2.3 Cas d'utilisation «Configurer la solution»	16
Chapitre 4 : Conception	18
4.1 Diagramme de classes de conception	18
4.2 Diagramme de séquence de conception.....	19
Chapitre 5 : Développement de la solution	22
5.1 Conception de l'API	22
5.1.1 Qu'est-ce qu'OpenAPI ?	23
5.1.2 Qu'est-ce que Swagger ?.....	23
5.1.3 La spécification OpenAPI de notre service web	23
5.2 La documentation de notre API.....	26
5.3 Extraction du texte.....	27
5.4 Classification de la facture.....	30
5.5 Extraction du montant	31
5.6 Extraction de la date	32
5.7 Déploiement de la solution sur Google Cloud Platform.....	32
5.7.1 Google Cloud Platform	33
5.7.2 Architecture applicative	35
5.7.3 Déploiement de la solution	39
5.8 Choix technique	44
5.9 Aperçu des résultats obtenus	45
Conclusion et perspectives	49
Bibliographie et Webographie	50

Liste des Figures

Figure 1 : Le développement de la société REDLEAN	4
Figure 2 : L'organisation de la société REDLEAN	4
Figure 3 : Architecture d'applications de types microservices via des API.....	7
Figure 4 : Relation typique client-serveur sur le web	8
Figure 5 : Logo de GCP	11
Figure 6 : Logo d'Expensya	11
Figure 7 : Logo de N2F.....	12
Figure 8 : Logo de Rydoo Expense.....	12
Figure 9 : Diagramme des cas d'utilisation générale	15
Figure 10 : Diagramme de séquence système du cas d'utilisation «Traiter une note de frais»	16
Figure 11 : Diagramme de séquence système du cas d'utilisation «Configurer la solution» .	17
Figure 12 : Diagramme de classes de conception	19
Figure 13 : Diagramme de séquence de conception pour l'opération système	20
«chargernotedeFrais(file)»	
Figure 14 : Diagramme de séquence de conception pour l'opération système «valider».....	21
Figure 15 : Passage automatique de la conception à la documentation d'une API.....	22
Figure 16 : Spécification OpenAPI de notre service web	25
Figure 17 : La suite de la spécification OpenAPI	26
Figure 18 : La documentation de service GET de notre API.....	26
Figure 19 : La documentation de service POST de notre API	27
Figure 20 : La suite de la documentation de service POST de notre API.....	27
Figure 21 : Extraction du texte	30
Figure 22 : Fichier de classification de la facture	31
Figure 23 : Les cas d'utilisation du Cloud Functions et App Engine	33
Figure 24 : Logo de Google App Engine	34
Figure 25 : Logo de Google Cloud Functions.....	34
Figure 26 : Les déclencheurs d'une fonction Cloud	35
Figure 27 : Architecture applicative.....	36
Figure 28 : Structure de Cloud Storage.....	37
Figure 29 : Relations éditeur-abonné	38
Figure 30 : Fenêtre de déploiement de la fonction extraction.....	40

Figure 31 : Fenêtre de déploiement de la fonction classification.....	41
Figure 32 : Fenêtre de déploiement de la fonction textMining	41
Figure 33 : La configuration de notre service web.....	42
Figure 34 : Tarifs de produit App Engine	43
Figure 35 : Les quotas gratuits de produit App Engine.....	43
Figure 36 : Serveur Node.js VS Serveur classique	44
Figure 37 : Résultat du notre solution	46
Figure 38 : Résultat du notre solution	47
Figure 39 : Interface de configuration.....	48

Introduction générale

La note de frais permet au salarié de recenser chaque mois les dépenses professionnelles supportées pour l'exercice de son activité salariale. Elle est remise à l'employeur pour obtenir le remboursement des frais professionnels.

La gestion des notes de frais constitue un véritable casse-tête pour les entreprises. Dans cette perspective, L'entreprise Redlean cherche à atteindre ces 2 objectifs :

- Supprimer la saisie des notes de frais fastidieuses et répétitives,
- Digitaliser les dépenses : Le collaborateur prend en photo ses dépenses et le système va automatiquement en reconnaître ce qu'il y a dedans et faire tout le flux jusqu'à l'expert-comptable.

C'est dans cet esprit que s'inscrit notre stage de fin d'études, intitulé «Création d'une solution intelligente pour l'automatisation de génération des dépenses», dont l'objectif est de concevoir et réaliser une application permettant générer automatiquement les dépenses.

Pour ce faire, nous procédons par une étude théorique afin de mieux cerner le contexte de notre travail.

Cette étude fait partie des objectifs de notre rapport qui est subdivisé en cinq chapitres :

- Le premier est consacré à la mise en relief du contexte générale de notre projet en présentant l'organisme d'accueil, la problématique traitée, les objectifs visés ainsi la conduite du projet adoptée pour le déroulement du stage,
- Dans le second chapitre, intitulé «Etude préalable», nous présentons les concepts théoriques utiles à l'élaboration de notre projet ainsi une étude des solutions similaires à notre solution proposée,
- Quant au troisième chapitre, intitulé «Analyse et spécification des besoins», nous identifions les besoins fonctionnels et non fonctionnels auxquels doit répondre notre application, en les modélisant à travers le diagramme des cas d'utilisation et le diagramme de séquence système,
- Le quatrième chapitre, il porte une démonstration de la conception adoptée pour répondre aux besoins précédemment cités,

- Finalement dans le dernier chapitre, nous expliquons toutes les étapes de réalisation de notre projet, ainsi nous présentons notre choix technologique et quelques aperçus des résultats obtenues.

Chapitre 1 : Contexte générale

Introduction

Dans ce premier chapitre, nous présentons d'une manière générale le contexte de travail et les objectifs de notre projet. Nous commençons par spécifier le cadre du projet puis nous présentons Redlean comme étant l'organisme d'accueil. Ensuite, nous présentons notre projet en passant par décrire le contexte initial, faire une étude et critique de l'existant et expliquer la solution envisagée. Enfin, nous précisons le choix méthodologique et nous présentons une planification possible du projet.

1.1 Cadre du projet

Dans le cadre de la formation d'ingénieur en génie logiciel au sein de l'Institut Supérieur des Sciences Appliquées et de Technologie de Sousse (ISSAT Sousse), nous terminons notre formation par un projet de fin d'études intitulé «Création d'une solution intelligente pour l'automatisation de génération des dépenses» qui met en pratique les connaissances techniques que nous avons acqui pendant notre formation académique. Pour guider ce modeste travail à terme dans une durée de cinq mois, nous avons été accueilli au sein de la société Redlean.

1.2 Organisme d'accueil : Redlean services

Redlean services, basée à Monastir, est spécialisée dans le développement des solutions informatiques innovantes. Son objectif est de concevoir et mettre en œuvre les meilleures solutions technologiques visant à améliorer la productivité, la rentabilité et la réactivité des entreprises sur le marché. En suivant une approche minimaliste et en appliquant les meilleures pratiques, les «redleaners» offrent des solutions de haute qualité qui améliorent notre vie numérique et évoluent nos habitudes. Redlean est une startup, fondée en 2016. Elle a connu une croissance remarquable ces dernières années.

La figure 1 représente le développement de la société REDLEAN.

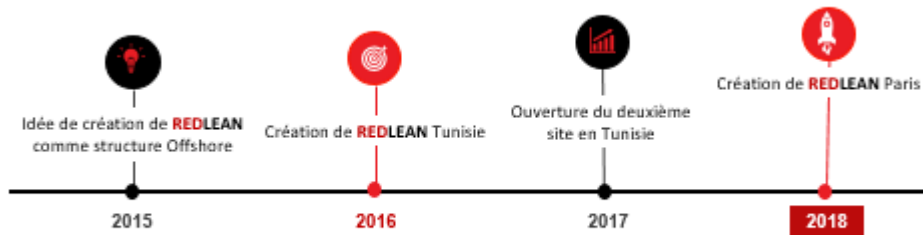


Figure 1 : Le développement de la société REDLEAN

1.2.1 Département d'accueil : Nouvelles technologiques

Redlean regroupe 4 principales unités comme suit :

- Equipe «Nouvelles Technologiques» qui se charge de développer des applications web et mobile, des API et des solutions de commerce électronique,
- Equipe «Performance Applicative» qui a pour mission d'assurer l'audit de code, audit d'architecture, tests de performance, analyse de problème de performances, etc,
- Equipe «Data» qui se charge de développer des solutions innovantes dans le domaine Datamining, Big Data, Data Science, etc,
- Equipe «Salesforce» qui a pour mission de développer des composants lightening, de créer des metadatas, des rapports, en plus d'assurer l'intégration des données.

La figure 2 représente l'organisation de la société REDLEAN.

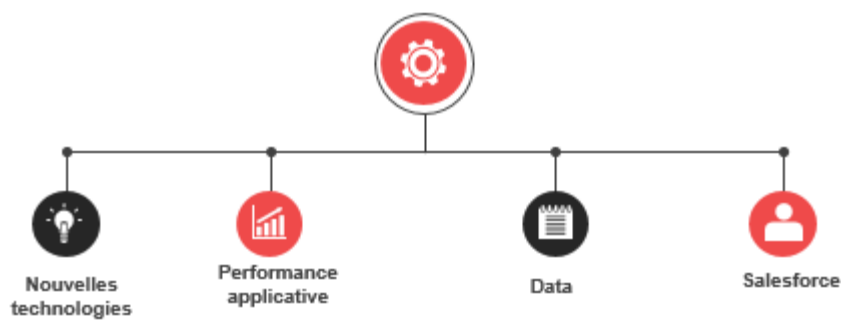


Figure 2 : L'organisation de la société REDLEAN

1.3 Etude de l'existant

1.3.1 Description de l'existant

Au sein de la société Redlean, à la fin de chaque mois, les collaborateurs seront invités à saisir manuellement ces notes de frais sur l'outil BoondManager.

Après avoir fini la saisie de ces notes de frais, le collaborateur valide le document puis un mail sera envoyé au validateur suivant.

1.3.2 Critique de l'existant

BoondManager est une solution de gestion en ligne intuitive, rapide à prendre en main et simple à utiliser au quotidien pour tous les profils métiers : managers, ingénieurs d'affaires, chargés de recrutement, collaborateurs, mais avec cet outil, chaque mois, chaque salarié passe entre 2 et 4 heures à traiter ces notes de frais et cela résulte une perte réelle.

1.3.3 Solution proposée

Pour résoudre cette problématique, nous proposons la création d'un module d'intelligence artificielle permettant l'automatisation de la génération des dépenses.

Le collaborateur Redlean prend en photo ses dépenses et le système va automatiquement en reconnaître ce qu'il y a dedans.

Ce module est représenté en une API Rest pour permettre d'utiliser ce traitement dans n'importe quel type d'application (web, mobile, desktop, ...). En plus nous avons déployé ce service web sur un fournisseur Cloud (Google Cloud Platform) pour bénéficier d'une architecture Serverless.

1.4 Organisation

1.4.1 Méthode agile

Dans le cadre de la gestion de projet, l'entreprise Redlean a fait le choix de réunions «Scrum» tous les 15 jours. Nous réunissons pour partager l'état d'avancement du projet et signaler les obstacles rencontrés. Nous considérons ces réunions comme l'occasion d'échanger les idées et pas uniquement de suivre la progression du projet.

1.4.2 Planification

Pour atteindre l'objectif fixé, la planification est une phase indispensable. En effet, la planification a pour objectif d'organiser le déroulement des étapes du projet dans le temps. Pour cela, nous avons découpé notre projet en plusieurs étapes.

Le tableau suivant représente la planification des tâches de notre projet.

Tableau 2 : Planification du projet

Taches	Février	Mars	Avril	Mai	Juin	Juillet
Etude des concepts clés						
Développement de l'API						
Ecrire les algorithmes nécessaires de la solution						
Déploiement de la solution						
Rapport						

Conclusion

Tout au long de ce chapitre, nous avons spécifié le cadre du projet puis nous avons présenté l'organisme d'accueil à savoir son développement et son organisation. Ensuite nous avons présenté notre projet, son contexte initial, une étude et critique de l'existant et une explication de la solution. Enfin, nous avons présenté la méthodologie de travail et la planification de notre projet.

Chapitre 2 : Etude préalable

Introduction

Dans ce deuxième chapitre, nous proposons une étude des concepts clés, puis nous étudions quelques solutions similaires.

2.1 Etude des concepts clés

Pour mieux comprendre le projet, nous consacrons cette partie pour la présentation des concepts clés liés à notre application intitulée «Création d'une solution intelligente pour l'automatisation de génération des dépenses».

— API

API signifie Application Programming Interface. Elle permet d'exposer certaines fonctionnalités et les rendre accessibles aux développeurs. Une API n'est pas destinée aux utilisateurs, mais pour les machines. On appelle aussi une API, un service web.

La figure 3 représente une architecture d'applications de types microservices via des API.

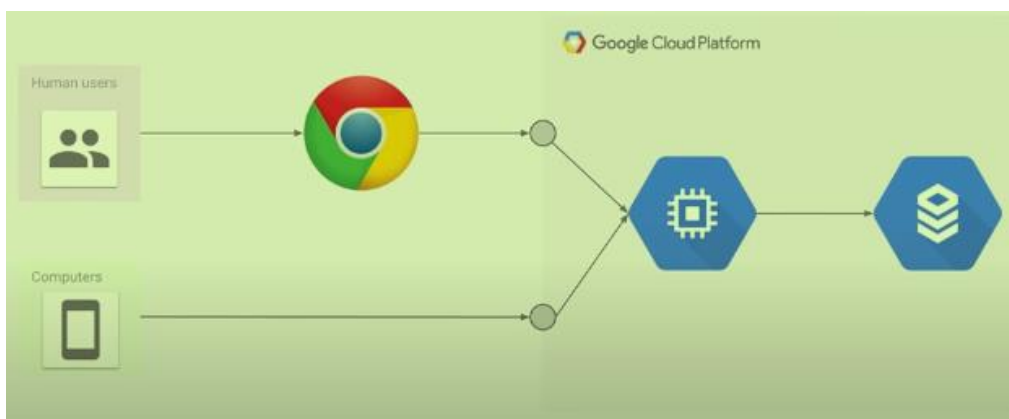


Figure 3 : Architecture d'applications de types microservices via des API

— API REST

C'est une API qui se base sur le protocole HTTP et ça donne l'avantage de pouvoir l'utiliser en programmation web mais aussi en programmation mobile.

— Les critères REST

REST signifie «Representational State Transfer». Les API REST imitent la façon dont le web lui-même marche dans les échanges entre un client et un serveur. Une API REST est sans état et orienté client-serveur. Le principe du client-serveur définit les deux entités qui interagissent dans une API REST : un client et un serveur, les mêmes entités qui communiquent sur le web. Un client envoie une requête, et le serveur renvoie une réponse. Ce dernier doit avoir le plus d'informations possible sur le client, car il est important qu'ils soient capables de travailler indépendamment l'un de l'autre.

La figure 4 représente une relation typique client-serveur sur le web.

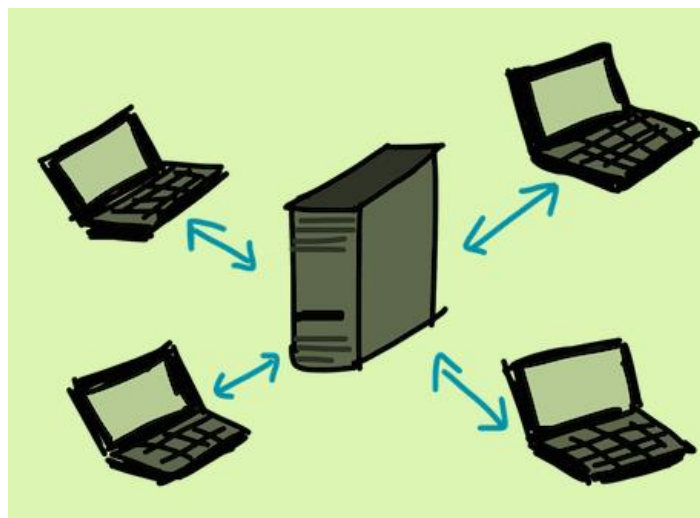


Figure 4 : Relation typique client-serveur sur le web

Le fait d'être «sans état» signifie que le serveur n'a aucune idée de l'état du client entre deux requêtes. Du point de vue du serveur, chaque requête est une entité distincte des autres.

Les réponses du serveur pour les API REST peuvent être délivrées dans de multiples formats. JSON (JavaScript Object Notation) est souvent utilisé, mais XML, CSV, ou même RSS sont aussi valables.

— Serverless

On a l'habitude de développer et de déployer des applications web où on a le contrôle sur les requêtes HTTP entrantes sur nos serveurs. Ces applications tournent sur des serveurs et

on est responsable de provisionner et de manager leurs ressources, ce qui peut poser plusieurs problèmes.

- On doit maintenir les serveurs disponibles même lorsqu'il n'y a pas de requêtes à traiter,
- On est responsable de la disponibilité et de la maintenance des serveurs et de leurs ressources,
- On doit ajuster les serveurs avec la charge : augmenter lorsque la charge arrive et diminuer lorsque la charge redescend.

Cela peut être très difficile à gérer pour les développeurs individuels. Cela finit par nous éloigner de notre mission initiale : construire et maintenir des applications au quotidien. Cela relève le plus souvent de la responsabilité de l'équipe infrastructure et rarement des développeurs. Cependant, les processus nécessaires pour les supporter peuvent ralentir les développements. On ne peut pas développer d'application sans l'aide de l'équipe infrastructure. En tant que développeurs, on recherche une solution à ces problèmes et c'est là que le Serverless entre en jeu.

— **L'architecture Serverless**

L'architecture Serverless est un modèle dans lequel le fournisseur de services Cloud (AWS, Azure ou Google Cloud) est responsable de l'exécution d'un morceau de code en allouant de manière dynamique les ressources. Et il ne facture que la quantité de ressources utilisées pour exécuter le code. Le code est généralement exécuté dans des conteneurs sans état pouvant être déclenchés par divers événements, notamment des requêtes HTTP, des événements de base de données, des services de file d'attente, des alertes de surveillance, des téléchargements de fichiers, des événements planifiés, etc. Le code envoyé au fournisseur de Cloud pour l'exécution est généralement sous la forme d'une fonction. Par conséquent, Serverless est parfois appelé «Functions as a Service» ou «FaaS». Alors que le Serverless isole l'infrastructure sous-jacente du développeur, les serveurs sont toujours impliqués dans l'exécution de nos fonctions. Étant donné que notre code va être exécuté en tant que fonctions individuelles, nous devons être conscients de certaines choses.

— **Microservices**

Le plus grand changement auquel on est confrontés lors de la transition vers un monde Serverless est que notre application doit être structurée sous forme de fonctions. Nous avons

peut-être l'habitude de déployer des applications monolithiques. Mais dans le monde Serverless, on doit généralement adopter une architecture davantage basée sur les microservices. Nous pouvons contourner ce problème en exécutant l'intégralité de notre application dans une seule fonction en tant que monolithe et en gérant nous-même le routage. Mais ceci n'est pas recommandé car il est préférable de réduire la taille de nos fonctions.

— Fonctions sans état

Les fonctions sont généralement exécutées dans des conteneurs sécurisés sans état (stateless). Cela signifie qu'on ne peut pas exécuter de code sur les serveurs d'applications, qui s'exécute longtemps après la fin d'un événement ou qui utilise le précédent contexte d'exécution pour répondre à une requête. On doit effectivement supposer que notre fonction est à nouveau invoquée à chaque fois.

— Google Cloud Platform

Google Cloud Platform, proposé par Google, est une suite de services de Cloud computing qui s'exécute sur la même infrastructure que Google utilise en interne pour ses produits destinés aux utilisateurs finaux, tels que Google Search, Gmail, file storage et Youtube.

L'infrastructure du Google est caractérisée par 3 types de puissance :

- Puissance de communication,
- Puissance de stockage,
- Puissance de calcul.

Il se trouve que cette infrastructure que Google a mis en œuvre pour ces propres besoins, la propose maintenant à l'allocation. Donc quand nous partons sur le Cloud de Google, nous allons louer l'infrastructure qui a été imaginé par les ingénieurs de Google pour les développeurs de Google.

Ainsi, Google Cloud Platform offre une technologie qui permet de réaliser des projets de hautes qualités. En plus, Elle permet de réduire les risques grâce à une sécurité de pointe.



Figure 5 : Logo de GCP

2.2 Etude des solutions similaires

Cette partie est consacrée pour présenter quelques solutions similaires à notre solution proposée. Nous présentons dans ce qui suit les trois solutions les plus reconnues et utilisées pour ce genre de besoin.

2.2.1 Expensya

C'est une solution accessible depuis web et mobile. Elle permet de numériser le justificatif pour en extraire les informations nécessaires puis les envoie vers la chaîne de comptabilité grâce à des technologies comme l'intelligence artificielle, machine Learning et stockage Cloud.



Figure 6 : Logo d'Expensya

2.2.2 N2F

Le scan intelligent de N2F extrait des données à partir de la photo en utilisant la machine Learning pour aboutir à des résultats exceptionnels en temps de traitement, des notes de frais se remplissent automatiquement sans délai ! Ce logiciel est interconnecté avec la majorité des banques et agences de voyage européenne. Il se caractérise par une grande fluidité dans son utilisation et par souplesse de paramétrage pour s'adapter facilement à tout type d'entreprises.



Figure 7 : Logo de N2F

2.2.3 Rydoo Expense

C'est une solution dématérialisée de gestion de frais sur smartphones, tablettes et ordinateurs, disponible sur iOS et Android. Cette application permet de gérer les dépendances en temps réel et de mettre fin aux interminables notes de frais. Son fonctionnement est simple : afin de se faire rembourser, il suffit de prendre en photo le justificatif de dépense qui sera contrôlé puis validé par l'équipe comptable.



Figure 8 : Logo de Rydoo Expense

Conclusion

Tout au long de ce chapitre, nous avons spécifié les différents concepts clés qui présentent la base de notre projet. Nous avons aussi étudié les autres solutions déjà existantes et similaires à notre solution proposée.

Chapitre 3 : Analyse et spécification des besoins

Introduction

Après avoir exposé le cadre générale de notre projet et l'étude préalable, on s'intéresse dans ce chapitre à l'identification des fonctionnalités de l'application qu'on envisage mettre en place. Ce chapitre représente la phase d'analyse des besoins du projet. C'est une étape déterminante avant d'entamer la phase de spécialisation conceptuelle. Elle permet une bonne compréhension des besoins et facilite l'identification des fonctionnalités demandées d'une façon pertinente à travers l'exposition des diagrammes des cas d'utilisation et de séquence système.

3.1 Analyse des besoins

Cette partie vise à spécifier l'objectif de ce projet en terme métier et le processus visant à établir les fonctionnalités que le système doit fournir et les contraintes auxquelles il sera soumis. C'est ce qu'on décrit ci-après.

3.1.1 Identification des acteurs

Un acteur est une entité externe qui interagisse directement avec le système. En fonction de ses actions, le système lui fournit le service adéquat à son besoin en question. Notre solution s'adresse à deux acteurs comme suit :

- L'employé Redlean qui utilise notre système pour traiter sa note de frais,
- L'administrateur qui permet de configurer la solution en fonction des besoins précis de la société Redlean.

3.1.2 Besoins fonctionnels

Les besoins fonctionnels sont ceux qui doivent répondre aux exigences de la solution en termes de fonctionnalité.

Notre système doit être capable d'analyser différents types de notes de frais afin de les classer et de récupérer l'ensemble des informations nécessaires. Les différentes classifications sont les suivantes :

- Restaurant,
- Hôtel,
- Train,
- Avion,
- Parking,
- Péage.

Une fois la classification réalisée, notre solution doit savoir extraire les informations clés, à savoir :

- Montant TTC,
- Date.

Notre solution s'intéresse aux notes de frais écrites en langue française.

3.1.3 Besoins non-fonctionnels

Les besoins non fonctionnels sont les contraintes que notre système doit respecter pour établir un bon fonctionnement. Les principaux besoins non fonctionnels de notre solution se résument dans les points suivants :

- Performance : Notre système doit avoir un temps de réponse raisonnable,
- Maintenabilité et évolutivité : Le code de l'application doit être bien lisible et compréhensible pour pouvoir le maintenir facilement et rapidement,
- Scalabilité : Notre solution doit supporter les fortes montées en charges.

3.2 Analyse semi-formelle des besoins

3.2.1 Spécification des besoins

Avant de toucher la partie de la conception, une étude plus approfondie des besoins fonctionnels qui nous aide à comprendre les besoins de chaque utilisateur s'avère indispensable. Nous présentons donc ici une spécification plus détaillée des besoins en s'appuyant sur les concepts de la modélisation UML.

3.2.1.1 Diagramme des cas d'utilisation

Le diagramme des cas d'utilisation permet l'identification des principales fonctionnalités à offrir par acteur. D'une façon générale, l'agent Redlean utilise notre système

pour traiter ses notes de frais. Il va oublier la saisie de ces frais professionnels et laisse notre solution faire pour une meilleur gestion de notes de frais. Ainsi, l'administrateur va configurer la solution en fonction des besoins précis de la société Redlean.

Nous présentons dans la figure ci-dessous le diagramme des cas d'utilisation générale qui englobe les fonctionnalités principales de notre système.

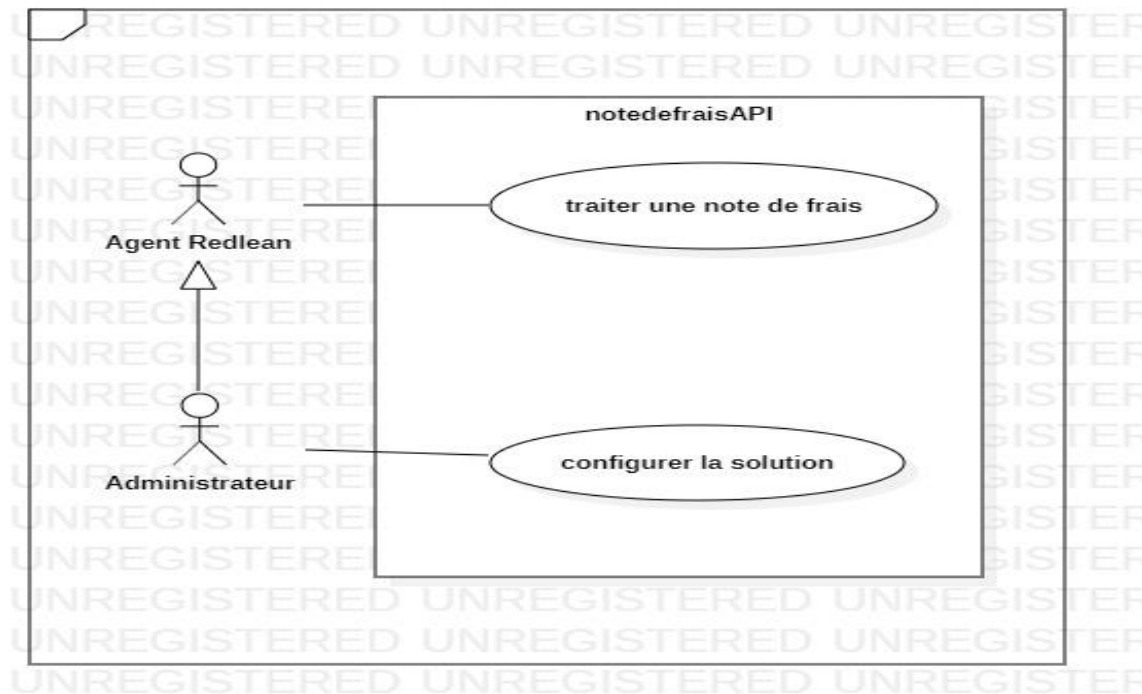


Figure 9 : Diagramme des cas d'utilisation générale

3.2.2 Cas d'utilisation «Traiter une note de frais»

On va choisir dans cette sous-partie de présenter le diagramme de séquence système pour le cas d'utilisation «Traiter une note de frais». Le diagramme de séquence système permet de décrire le scénario nominal d'un cas d'utilisation en considérant le système comme une boîte noire. L'agent Redlean va charger un fichier image ou PDF dans notre système ensuite ce dernier va renvoyer la réponse qui contient le montant TTC, la date et la catégorie de la dépense.

La figure 10 représente le diagramme de séquence système du scénario nominale du cas d'utilisation «Traiter une note de frais».

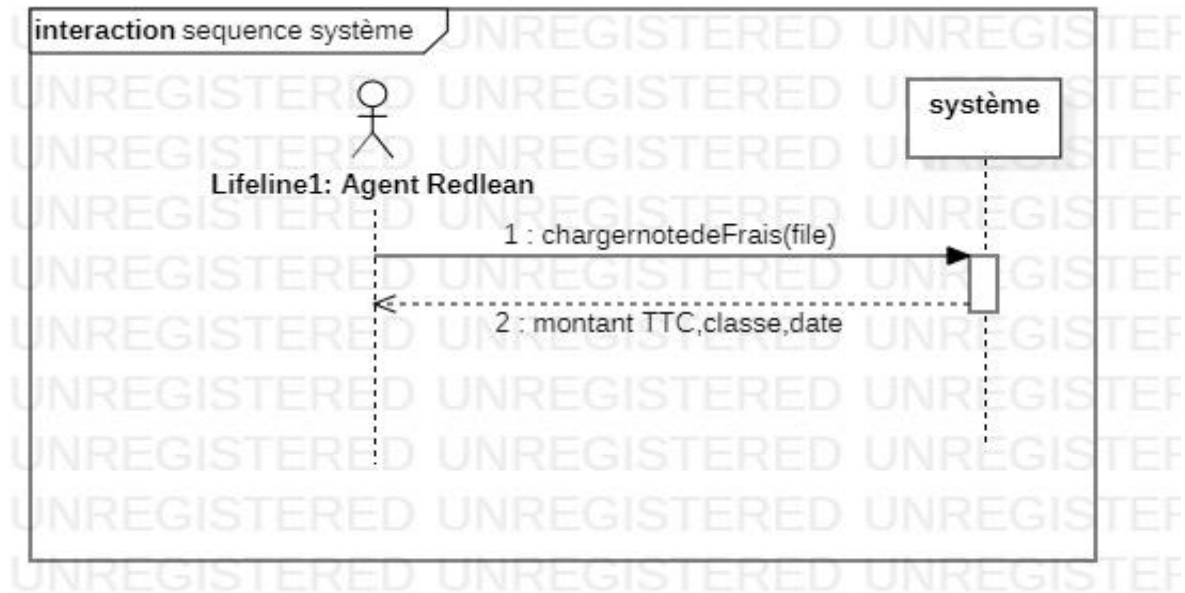


Figure 10 : Diagramme de séquence système du cas d'utilisation «Traiter une note de frais»

3.2.3 Cas d'utilisation «Configurer la solution»

On va choisir dans cette sous-partie de présenter le diagramme de séquence système pour le cas d'utilisation «Configurer la solution». Selon les besoins de la société Redlean, l'administrateur de l'application va saisir dans le système un ensemble de mots clés qui définissent les différentes classifications considérées.

La figure 11 représente le diagramme de séquence système du scénario nominale du cas d'utilisation «Configurer la solution».

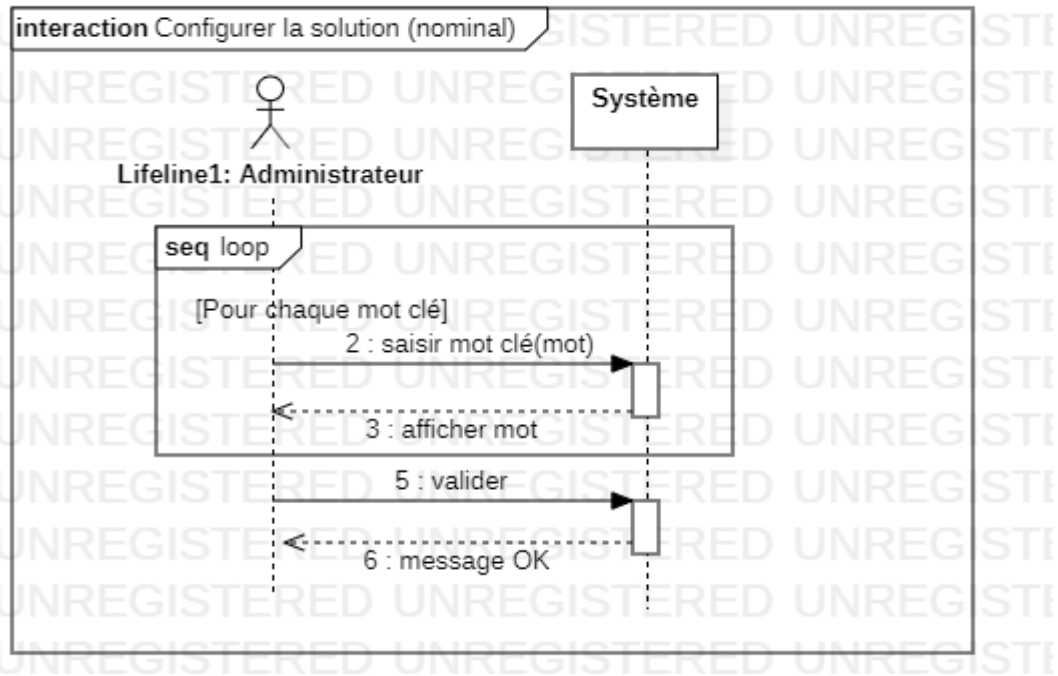


Figure 11 : Diagramme de séquence système du cas d'utilisation «Configurer la solution»

Conclusion

Dans ce chapitre, nous avons décelé les besoins fonctionnels et non fonctionnels ainsi que les acteurs de notre système. Par la suite, nous avons eu recours aux diagrammes de cas d'utilisation et séquence système pour détailler les besoins. Dans le chapitre suivant, nous décrivons la conception détaillée du système.

Chapitre 4 : Conception

Introduction

La conception étant une phase cruciale dans le processus de réalisation d'un projet. Elle s'agit de la manière à implémenter le module.

Suite aux spécifications fonctionnelles et non fonctionnelles, nous commençons la phase de l'élaboration de la conception. Dans ce chapitre, nous avons présenté le diagramme de classes de conception et le diagramme de séquence de conception de notre application.

4.1 Diagramme de classes de conception

Dans cette sous-section, nous allons détailler le processus du traitement des notes de frais. La solution doit être capable d'analyser différents types de notes de frais afin de les classer et de récupérer l'ensemble des informations nécessaires. Les différentes classifications sont les suivantes :

- Restaurant,
- Hôtel,
- Train,
- Avion,
- Parking,
- Péage.

Une fois la classification est réalisée, un algorithme doit savoir extraire les informations clés, à savoir : montant TTC et la date. Pour ce faire, le diagramme de classe représenté par la figure 12 montre la structure algorithmique de la solution adoptée.

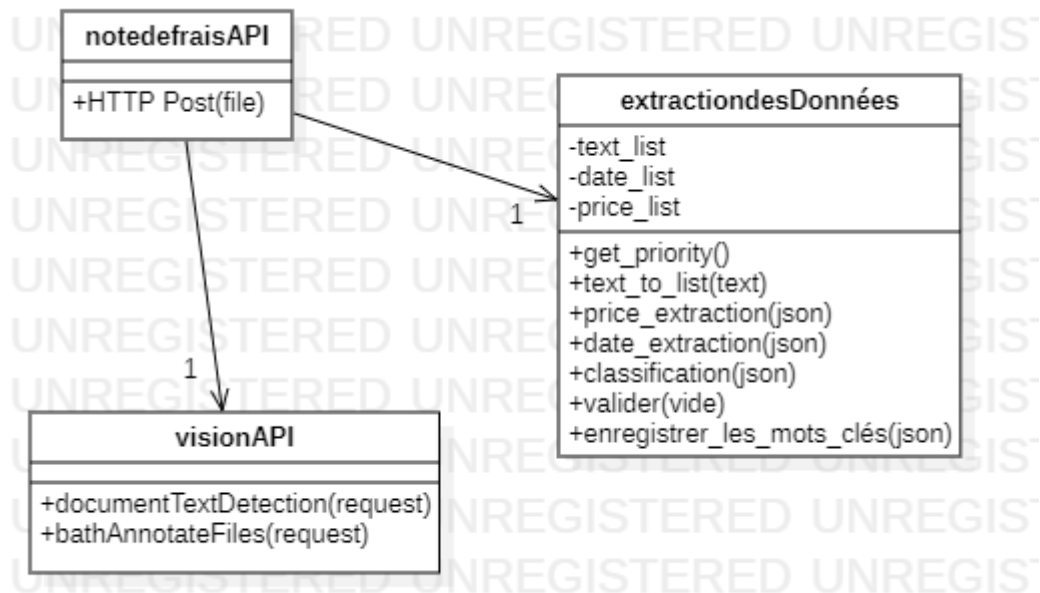


Figure 12 : Diagramme de classes de conception

- **notedefraisAPI** : Cette classe permet de gérer les requêtes HTTP. Elle permet de télécharger la note de frais dans notre application.
- **extractiondesDonnées** : Cette classe permet d'extraire les données envisageables à partir de l'image ou document PDF de note de frais.
 - `price_extraction(json)` et `date_extraction(json)` : permettent de déterminer respectivement le montant total et la date de la note de frais.
 - `classification(json)` : permet de déterminer à quelle classe, parmi les suivantes, appartient la note de frais : restaurant, péage, parking, hôtel, train, avion.
- **visionAPI** : Cette classe représente les fonctionnalités de l'API Vision utilisés pour l'extraction du texte de la note de frais dans notre solution.

4.2 Diagramme de séquence de conception

Dans cette sous-section, nous avons détaillé le fonctionnement interne de système. En premier lieu, nous avons détaillé l'opération système `chargernoteFrais(file)` pour bien expliqué la réalisation du cas d'utilisation «Traiter une note de frais». En deuxième lieu, nous avons détaillé l'opération système «valider» pour bien expliqué la réalisation du cas d'utilisation «Configurer la solution».

Les figures 13 et 14 représentent le diagramme de séquence de conception des opérations citées au niveau du paragraphe précédent.

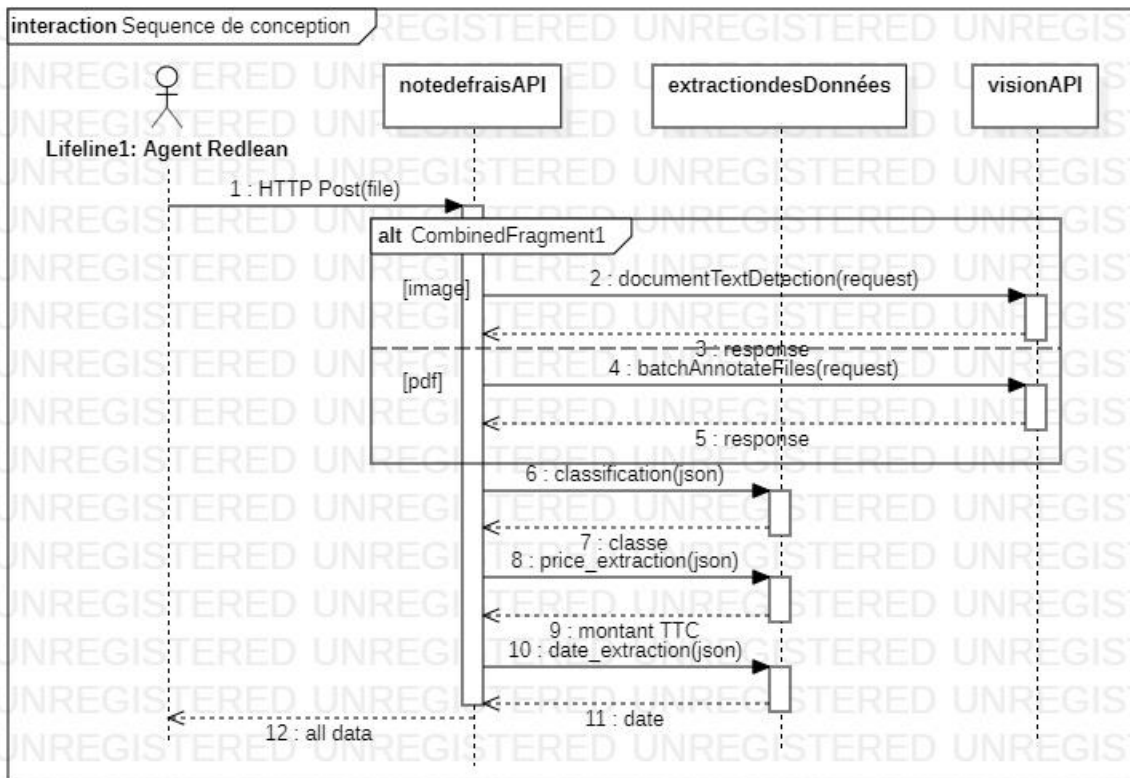


Figure 13 : Diagramme de séquence de conception pour l'opération système
«chargernoteFrais(file)»

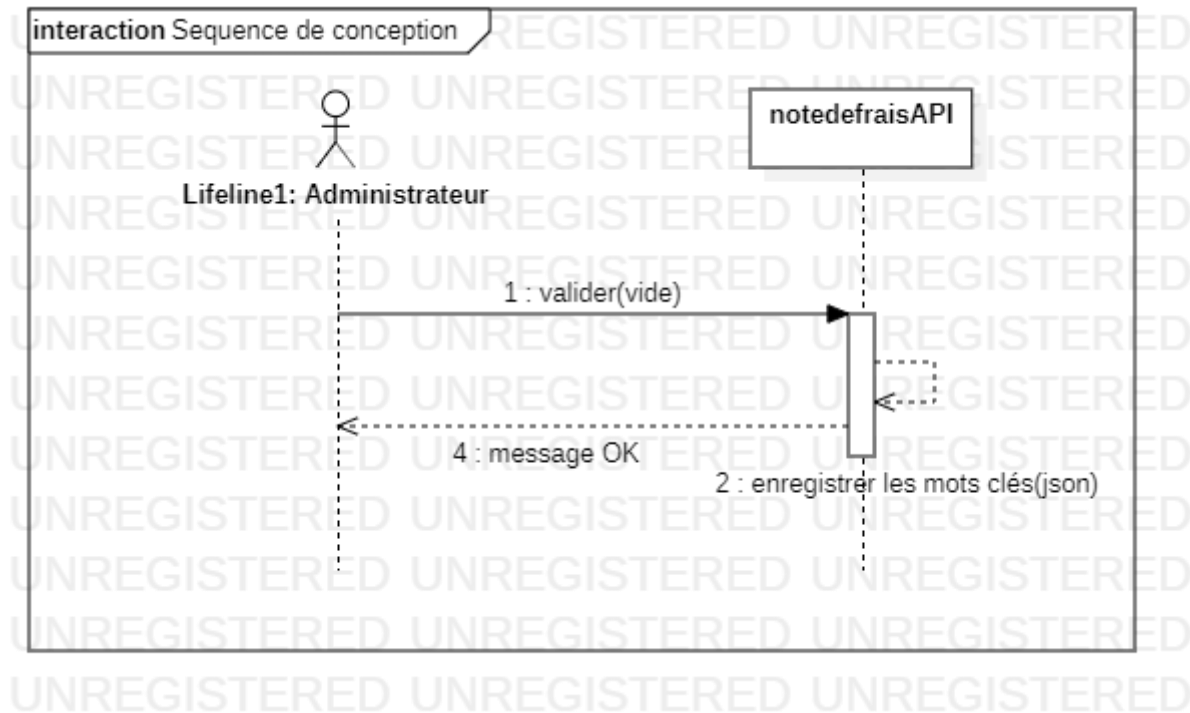


Figure 14 : Diagramme de séquence de conception pour l’opération système «valider»

Conclusion

Au cours de ce chapitre, nous avons mis en place la partie conception de l’application. Nous avons montré une vue statique du système. Ainsi, nous avons analysé son comportement dynamique. Dans le chapitre suivant, nous décrivons la partie réalisation de l’application.

Chapitre 5 : Développement de la solution

Introduction

Le présent chapitre décrit la partie développement de notre solution. Nous avons se concentré sur le traitement de la note de frais. Le processus du traitement commence par le téléchargement de la facture dans l'application via une API Rest. Ensuite, la deuxième phase consiste à extraire le texte de la note de frais. La troisième phase consiste à développer des algorithmes pour extraire la catégorie, le montant et la date de la note de frais. Enfin, nous avons déployé notre solution sur Google Cloud Platform. A la fin de ce chapitre, nous avons présenté notre choix technique et quelques aperçus des résultats obtenues.

5.1 Conception de l'API

La conception est la base de notre développement d'API. Les meilleurs API sont conçues en pensant au consommateur final. Swagger Editor a été le premier éditeur conçu pour la conception d'API avec la spécification OpenAPI. L'éditeur valide notre conception en temps réel, vérifie la conformité de l'OAS (OpenAPI Specification) et fournit un retour visuel de la documentation de l'API.

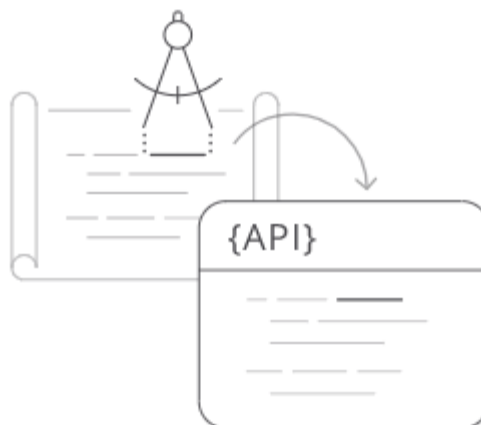


Figure 15 : Passage automatique de la conception à la documentation d'une API

5.1.1 Qu'est-ce qu'OpenAPI ?

La spécification OpenAPI (anciennement la spécification Swagger) est un format de description d'API pour les API REST.

Un fichier OpenAPI nous permet de décrire l'ensemble de notre API, y compris:

- Points de terminaison disponibles (ex : /users) et opérations (ex : GET /users, POST /users),
- Paramètres de fonctionnement entré et sortie pour chaque opération,
- Méthodes d'authentification,
- Coordonnées, licence, conditions d'utilisation et autres informations.

Les spécifications d'API peuvent être écrites en YAML ou JSON. Le format est facile à apprendre et lisible pour les humains et les machines. La spécification OpenAPI complète peut être trouvée sur GitHub : Spécification OpenAPI 2.0.

5.1.2 Qu'est-ce que Swagger ?

Swagger est un ensemble d'outils open source construit autour de la spécification OpenAPI qui peut nous aider à concevoir, construire, documenter et consommer des API REST.

Les principaux outils Swagger comprennent:

- Swagger Editor : Editeur basé sur un navigateur où vous pouvez écrire des spécifications OpenAPI,
- Swagger UI : Rend les spécifications OpenAPI sous forme de documentation API interactive,
- Swagger Codegen : Génère des stubs de serveur et des bibliothèques clientes à partir d'une spécification OpenAPI.

5.1.3 La spécification OpenAPI de notre service web

Nous pouvons écrire des définitions OpenAPI en YAML ou JSON. Dans notre cas, nous utilisons YAML.

Chaque définition d'API doit inclure la version de la spécification OpenAPI sur laquelle cette définition est basée. La version OpenAPI définit la structure globale d'une définition

d'API, ce que nous pouvons documenter et comment nous la documentons. Comme indiqué dans la figure 16, OpenAPI 2.0 est la version utilisée.

La section info contient des informations sur l'API : titre, description (facultatif) et version. Le nom de notre API est `notedefraisAPI` et elle permet de générer automatiquement des dépenses.

Ensuite, le fichier OpenAPI contient la section host qui spécifie l'URL de base de notre API. Enfin, la section des chemins qui définit les points de terminaison individuels (chemins) dans notre API et les méthodes HTTP (opérations) prises en charge par ces points de terminaison. Notre API expose deux opérations :

- POST `/upload`,
- GET `/`.

Une définition d'opération comprend les paramètres, le corps de la demande (le cas échéant), les codes d'état de réponse possibles (tels que 200 OK ou 404 Not Found) et le contenu de la réponse.

```
1 swagger: "2.0"
2 info:
3   title: notedefraisAPI
4   description: API pour la génération automatique des dépenses.
5   version: 1.0.0
6 host: projet-pfe-278305.uc.r.appspot.com
7 schemes:
8   - https
9 paths:
10  /upload:
11    post:
12      summary: télécharger un fichier
13      consumes:
14        - multipart/form-data
15      produces:
16        - application/json
17      parameters:
18        - in: formData
19          name: file
20          type: file
21          description: Le fichier à télécharger
22      responses:
23        200:
24          description: OK
25          schema:
26            type: object
27            properties:
28              jsonresult:
29                type: array
30                items:
31                  type: object
32                  properties:
33                    classe:
34                      type: string
35                    date:
```

Figure 16 : Spécification OpenAPI de notre service web


```

31                                     type: object
32                                     properties:
33                                         classe:
34                                             type: string
35                                         date:
36                                             type: string
37                                         montant:
38                                             type: number
39                                     400:
40                                         description: aucune fichier téléchargée
41                                 /:
42                                 get:
43                                     summary: recevoir la page d'accueil
44                                     produces:
45                                         - text/html
46                                     responses:
47                                         200:
48                                             description: ok

```

Figure 17 : La suite de la spécification OpenAPI

5.2 La documentation de notre API

Notre API REST offre deux services. Le premier service permet de recevoir une page d'accueil sous format HTML et le deuxième service permet de télécharger une note de frais, ce dernier prend en paramètre un fichier image ou PDF et rendre une réponse JSON avec le modèle décrit dans la figure 20.

The screenshot shows the documentation for the GET endpoint `/ recevoir la page d'accueil`. It includes a 'Parameters' section with 'No parameters', a 'Responses' section with a 'Response content type' dropdown set to 'text/html', and a table of responses.

Code	Description
200	ok

Background text: Activer Windows. Accédez aux paramètres pour activer Windows.

Figure 18 : La documentation de service GET de notre API

POST

/upload télécharger un fichier

Parameters

Try it out

Name	Description
file	Le fichier à télécharger
file (formData)	<div>Choisir un fichier</div> <div>Aucun fichier choisi</div>

Responses

Response content type

application/json

Code	Description
200	OK

Example Value

Model

Figure 19 : La documentation de service POST de notre API

```

{
  "jsonresult": [
    {
      "classe": "string",
      "date": "string",
      "montant": 0
    }
  ]
}

```

400

aucune fichier téléchargée

Figure 20 : La suite de la documentation de service POST de notre API

5.3 Extraction du texte

Le traitement de la note de frais commence par l'extraction du texte à partir d'une image ou d'un document PDF.

L'extraction du texte d'une image de note de frais se fait à l'aide de la technique de reconnaissance optique de caractères, en anglais optical character recognition (OCR). La reconnaissance optique de caractères désigne les procédés informatiques pour la traduction d'images de textes imprimés ou dactylographiés en fichiers de texte. Dans notre cas, nous appliquons l'OCR à l'aide de l'API Google Cloud Vision. Les étapes de configuration nécessaires pour commencer à utiliser cette API sont les suivantes :

- Créer un projet : Pour utiliser les services fournis par GCP, nous devons créer un projet. Un projet permet d'organiser toutes nos ressources GCP. Un projet se compose des éléments suivants :
 - Un ensemble de collaborateurs,
 - Des API activées (et autres ressources),
 - Des outils de surveillance,
 - Des informations de facturation,
 - L'authentification et le contrôle des accès.
- Activer la facturation : Un compte de facturation sert à déterminer qui assume les frais pour un ensemble de ressources données. Les frais d'utilisation du projet sont imputés au compte de facturation associé. Nous configurons la facturation lorsque nous créons un projet,
- Activer l'API : Nous devons activer l'API Cloud Vision pour notre projet,
- Configurer l'authentification : Toute application cliente qui utilise l'API doit être authentifiée et se voir accorder l'accès aux ressources demandées. Les API Google Cloud n'acceptent que les requêtes provenant d'applications enregistrées, qui sont des applications identifiables de manière unique et qui présentent des informations d'identification au moment de la requête. Les requêtes provenant d'applications anonymes sont refusées. Dans notre cas, le type d'identifiant utilisé est la clé API. Notre travail consiste à créer une clé. Pour créer une clé API au sein d'un projet, l'utilisateur doit disposer du rôle primitif éditeur sur le projet.

Pour créer une clé API :

- Accédez au panneau API et services -----> Identifiants dans Cloud Console,
- Sélectionnez Créer des identifiants, puis sélectionnez Clé API dans le menu déroulant,

- La boîte de dialogue Clé API crée affiche la nouvelle clé. Nous devons peut-être copier la clé et la conserver à un emplacement sécurisé,
- Utiliser une clé API : Transmettez la clé API dans un appel d'API REST en tant que paramètre de requête. Remplacez API_Key par notre clé API.

L'API Vision peut détecter et extraire du texte à partir d'images. L'API comprend plusieurs fonctionnalités mais pour appliquer la reconnaissance optique des caractères dans notre application, nous utilisons la fonction `DOCUMENT_TEXT_DETECTION`. La réponse de cette fonction comprend des informations sur les blocs et les mots.

Pour le cas des documents PDF, l'API comprend deux méthodes pour l'extraction du texte. La première est une méthode asynchrone : une requête d'annotation hors connexion lance une opération de longue durée et ne renvoie pas immédiatement de réponse à l'appelant. Une fois l'opération de longue durée terminée, les annotations sont stockées sous forme de fichiers dans un bucket Cloud Storage que nous spécifions. La deuxième méthode est une méthode synchrone : une requête d'annotation en ligne renvoie immédiatement les annotations intégrées à l'utilisateur. La méthode asynchrone permet d'extraire le texte d'un fichier PDF de 2000 pages au maximum par contre la méthode synchrone permet d'extraire le texte d'un fichier PDF de 5 pages au maximum.

Après l'étude de ces deux méthodes, nous choisissons la deuxième (synchrone) parce qu'elle minimise le temps de réponse.

La figure 21 représente le résultat d'une extraction du texte à partir d'un justificatif du note de frais à l'aide de l'API Cloud Vision.



Figure 21 : Extraction du texte

A l'aide de la réponse du Cloud Vision API, on va essayer de classifier la facture et obtenir le prix et la date.

5.4 Classification de la facture

Notre travail consiste à développer un algorithme de classification de texte qui permet de catégoriser une note de frais dans l'une de plusieurs catégories. Dans notre cas, les différentes catégories sont les suivantes : Restaurant, Hôtel, Train, Avion, Parking, Péage.

Un fichier JSON contenant les classifications et les mots clés associés est nécessaire pour notre algorithme. Pour cela, nous avons développé une interface web qui permet de créer ce fichier JSON et l'ajouter à une base de données Firebase.

Notre algorithme de classification de texte consiste à parcourir le texte de la facture en cherchant les mots clés. Ensuite l'algorithme attribue un score à chaque classe et enfin choisit celle avec le score le plus élevé.

```
{
  "categories": {
    "names": [ "Restaurant", "Hôtel", "Train", "Avion", "Parking", "Péage" ],
    "topics": [ {
      "Restaurant": [ "café", "jus", "resto", "chicken", "pizza", "sandwich", "frite", "glace", "table", "couvert", "eau",
        | "burger", "food", "boisson", "plat", "repas", "restau", "dessert" ]
    }, {
      "Hôtel": [ "room", "chambre", "nuit", "hôtel", "reception", "étage", "repas", "appartement", "residence", "jour" ]
    }, {
      "Train": [ "train", "station", "arrivé", "métro", "classe", "depart", "gare", "banlieue", "cabine", "arrêt" ]
    }, {
      "Avion": [ "aéroport", "départ", "arrivé", "classe", "direct", "escale", "repas", "air", "vol", "passeport", "avion" ]
    }, {
      "Parking": [ "voiture", "matricule", "parking", "parc", "véhicule", "parc", "place", "stationnement" ]
    }, {
      "Péage": [ "péage", "véhicule", "autoroute" ]
    }
  ]
}
```

Figure 22 : Fichier de classification de la facture

5.5 Extraction du montant

L'extraction du total est la tâche la plus délicate. La réponse de l'API vision contient une séparation du bloc de chaque mot trouvé dans la facture dont chaque bloc est caractérisé par le mot détecté et les 4 couples de coordonnées (x,y) proportionnelle au taille du facture.

En se basant sur la réponse de l'API vision, notre solution était une combinaison de deux algorithmes. Le premier consiste à jouer sur les mots clés qui peuvent être associés au total de la facture. Nous indiquons à l'algorithme la liste des mots clés à recherchés.

Ensuite, l'algorithme doit tester la présence de ces mots clés dans le texte de la facture. Pour chaque mot clés trouvés, l'algorithme va concaténer toutes les mots se trouvant sur la même ligne (en utilisant l'axe y) et devant le mot clés recherché dans une chaîne de caractère et appliquera la fonction regex pour détecter le réel trouvant dans cette chaîne de caractère qui peut être la valeur du total. Chaque réel trouvé est stocké dans un tableau. Enfin, le total de la facture doit être le maximum de ce tableau.

Mais, si ce n'est pas le cas et la facture ne contient pas aucune des mots clés recherchés la solution passera à utiliser le deuxième algorithme.

La réponse de l'API vision comprend une représentation du texte sous forme des blocs. Le deuxième algorithme consiste à parcourir l'image bloc par bloc. L'algorithme consiste à éliminer les blocs qui contiennent les mots comme tva, t.v.a, tél, km, numéro afin d'assurer d'éliminer les réels qui indiquent le taux de tva, le nombre des kilomètres, le numéro de la facture,

ensuite d'extraire des blocs restants toutes les réels (en utilisant la fonction regex) dans la facture et choisir le maximum entre eux.

5.6 Extraction de la date

L'algorithme d'extraction de la date est basé sur les expressions régulières, souvent surnommées regex. Les regex sont une sorte de langage à part qui sert à manipuler les chaînes de caractères. Ils permettent d'extraire des informations d'une chaîne de caractères (bien plus puissant que de jouer avec `indexOf()` et `substring()`). L'algorithme parcourt la facture bloc par bloc et vérifie les règles spécifiées dans la regex.

Nous prenons en compte les formats des dates suivantes :

- JJ[-/]MM[-/]AA,
- JJ[-/]MM[-/]AAAA,
- JJ[-/]MM.

Cet algorithme est basé sur une logique conditionnelle.

5.7 Déploiement de la solution sur Google Cloud Platform

Nous choisissons d'adapter une architecture microservice sur notre solution. Les microservices désignent un style d'architecture utilisé dans le développement d'applications. Ils permettent de décomposer une application volumineuse en composants indépendants, chaque élément ayant ses propres responsabilités. Pour diffuser la requête d'une API, une application basée sur des microservices peut appeler plusieurs microservices internes pour composer sa réponse.

Une application basée sur des microservices correctement implémentée peut permettre d'atteindre les objectifs suivants :

- Clarifier les rapports de journalisation et de surveillance,
- Augmenter l'évolutivité et la fiabilité globales des applications,
- Etc.

Notre objectif est de séparer les fonctionnalités de notre application. Nous avons essayé de chercher un logique métier pouvant être séparé. Le résultat de la migration de notre application monolithique vers une application basée sur les microservices est la suivante :

- Un microservice nommée «notedefraisAPI» qui est responsable de la gestion des requêtes HTTP,
- Un microservice nommée «extraction_texte» qui permet d’extraire du texte d’une note de frais téléchargée dans notre application,
- Un microservice nommée «classification» qui permet d’accorder une classe à la note de frais,
- Un microservice nommée «text_mining» qui permet d’extraire le montant TTC et la date de la note de frais.

5.7.1 Google Cloud Platform

Google Cloud Platform fait partie d’un ensemble de solutions pour les entreprises appelé Google Cloud, et fournit des services modulaires basés sur le Cloud, tels que le stockage d’informations, le calcul, des applications de traduction et de prévision, etc.

Après des recherches sur cette plateforme, nous avons décidé d’utiliser les deux produits Google Cloud suivants : Google App Engine et Google Cloud Functions. La figure ci-dessous représente les cas d’utilisation de ces deux produits.

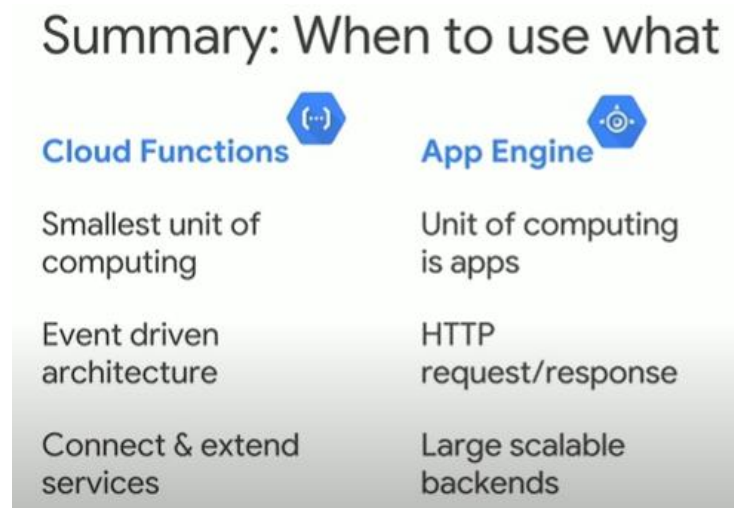


Figure 23 : Les cas d’utilisation du Cloud Functions et App Engine

5.7.1.1 Google App Engine

App Engine est une plate-forme sans serveur entièrement gérée pour le développement et l’hébergement d’applications Web à grande échelle. Nous pouvons choisir parmi plusieurs langages, bibliothèques et frameworks courant pour développer nos applications, puis laisser

App Engine se charge du provisionnement des serveurs et du scaling de nos instances d'applications en fonction de la demande.



Figure 24 : Logo de Google App Engine

5.7.1.2 Google Cloud Functions

Cloud Functions offre une expérience de développement simple et intuitive. Il nous suffit de rédiger notre code, et Google Cloud s'occupe de gérer l'infrastructure opérationnelle. Grâce à Cloud Functions, nous développons des solutions plus rapidement en créant et en exécutant des petits extraits de code qui répondent à des événements, et pour rationaliser les problèmes complexes d'orchestration, nous disposons de déclencheurs pour nous connecter à des services Cloud tiers ou à Google Cloud.



Figure 25 : Logo de Google Cloud Functions

La figure 26 représente les différents déclencheurs d'une fonction Cloud disponible pour l'utilisation dans la plateforme Google Cloud.

Déclencheur

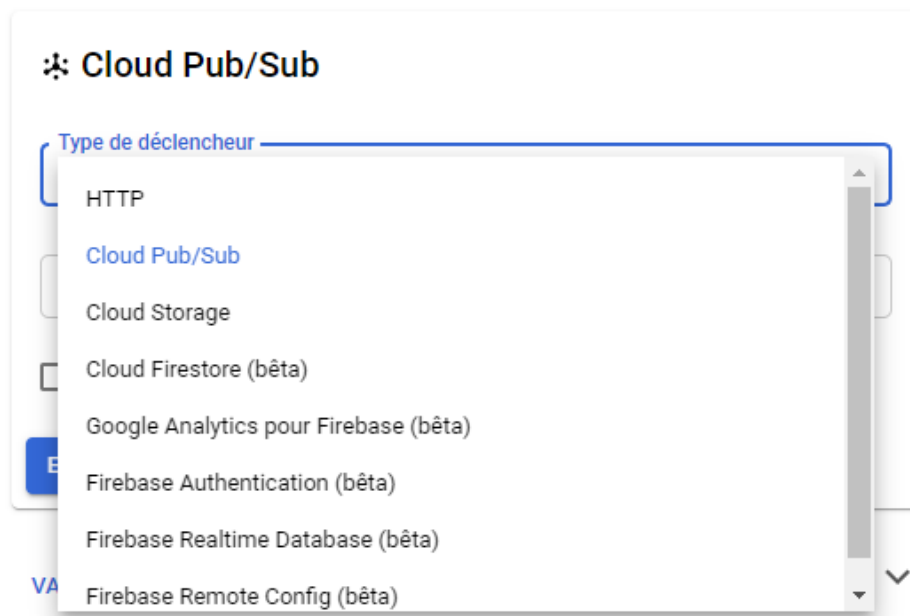


Figure 26 : Les déclencheurs d’une fonction Cloud

5.7.2 Architecture applicative

Pour déployer le service web `notedefraisAPI`, nous choisissons d’utiliser le produit Google App Engine. Concernant les autres microservices, nous choisissons le produit Google Cloud Functions. Pour assurer la bonne communication entre ces microservices, nous avons structuré notre solution comme suit :

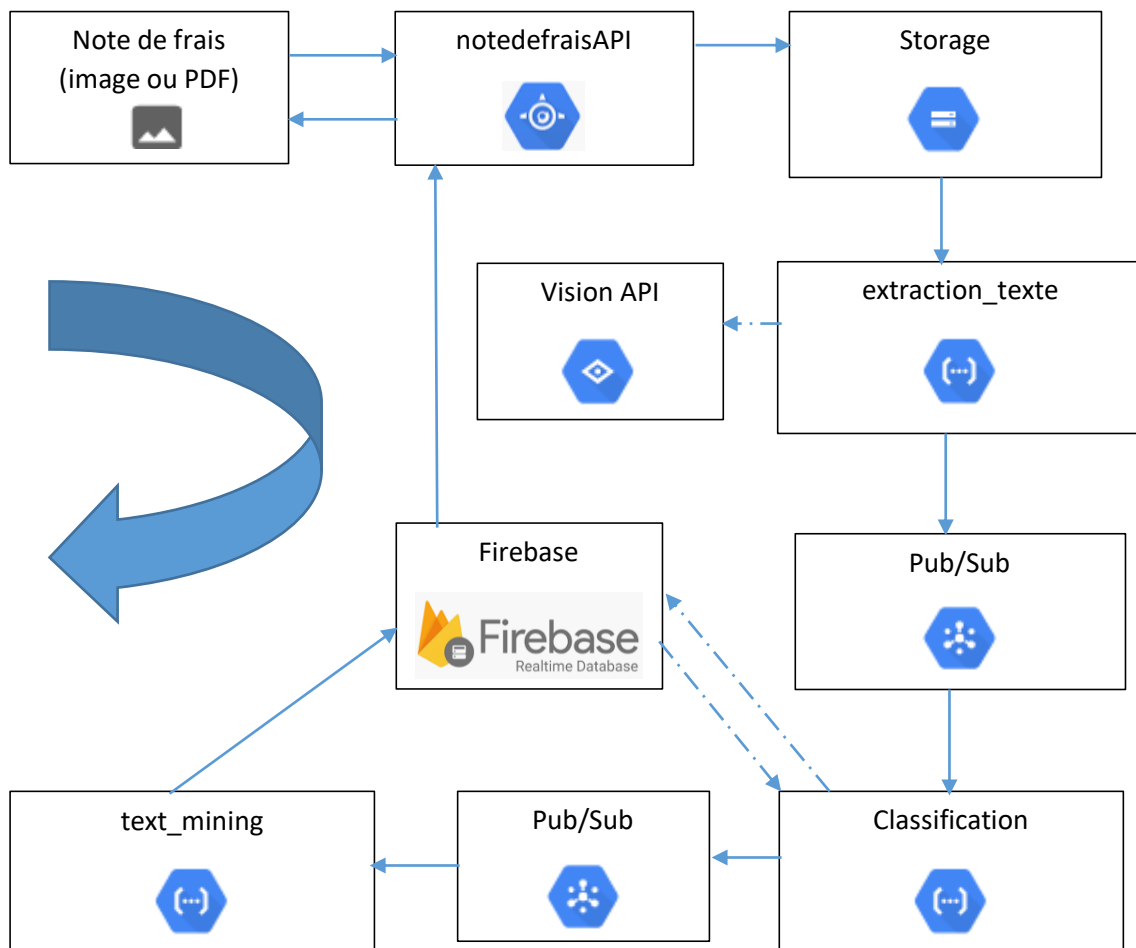


Figure 27 : Architecture applicative

5.7.2.1 Visualiser le flux de données

Le flux de données dans notre application comprend plusieurs étapes :

- Une note de frais (image ou PDF) en langue française est importée dans Cloud Storage,
- Une fonction Cloud utilisant l'API Vision pour extraire le texte est déclenchée,
- Un objet JSON contenant le texte est publié dans un sujet Pub/Sub,
- Une autre fonction Cloud est déclenchée qui permet de classer la note de frais. Cette fonction récupère de la base de données Firestore les mots clés utiles pour réaliser le traitement,
- Un objet JSON contenant le texte et la classe de la note de frais est publié dans un sujet Pub/Sub,

- Une autre fonction Cloud est déclenchée qui permet d'extraire le montant TTC et la date de la note de frais. Cette fonction permet aussi d'enregistrer le résultat final du traitement dans la base de données,
- Une fois le résultat est stocké dans la base de données, un évènement va être déclenché qui permet de transmettre un objet JSON contenant la classe, le montant et la date de la note de frais vers l'API.

5.7.2.2 Qu'est-ce que Cloud Storage ?

Cloud Storage est un service permettant de stocker nos objets dans Google Cloud. Un objet est une donnée immuable constituée d'un fichier qui peut être de n'importe quel format. Nous stockons des objets dans des conteneurs appelés buckets. Tous les buckets sont associés à un projet et nous pouvons regrouper nos projets dans une organisation.

Après avoir créé un projet, nous pouvons créer des buckets Cloud Storage, importer des objets dans nos buckets et télécharger des objets depuis nos buckets. Nous pouvons également accorder certaines autorisations pour rendre nos données accessibles aux membres que nous spécifions ou, dans certains cas d'utilisation tel que l'hébergement de site Web, les rendre accessibles à toutes les personnes se trouvant sur le réseau public.

La structure de Cloud Storage se présente comme suit :

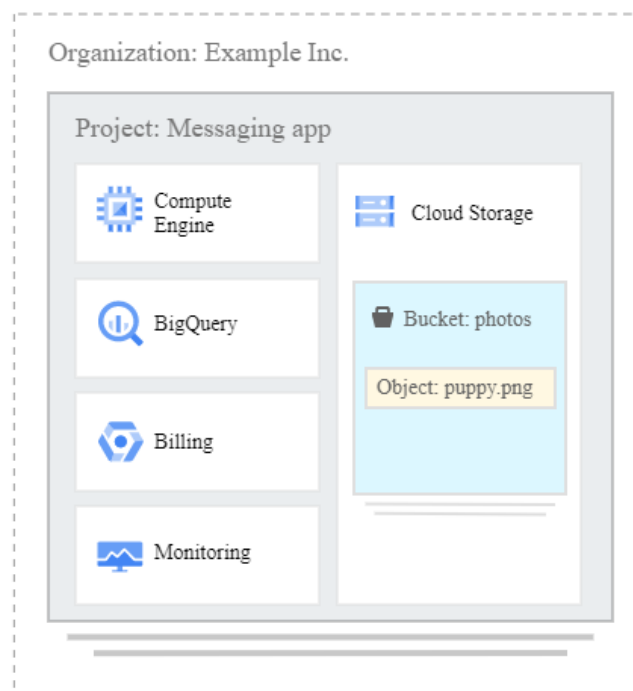


Figure 28 : Structure de Cloud Storage

5.7.2.3 Qu'est-ce que Pub/Sub ?

— Concepts fondamentaux

Pub/Sub est un service de messagerie asynchrone qui dissocie les services de production d'évènements des services de traitement des évènements. Nous pouvons utiliser Pub/Sub en tant que middleware de messagerie ou outil d'ingestion et de diffusion d'évènements pour les pipelines d'analyse en flux continu.

Pub/Sub offre un stockage durable des messages ainsi qu'une distribution des messages en temps réel, avec une haute disponibilité et des performances constantes à grande échelle. Les serveurs Pub/Sub fonctionnent dans toutes les régions Google Cloud réparties dans le monde entier.

— Relations éditeur-abonné

Une application d'éditeur crée et envoie des messages dans un sujet. Les applications d'abonnés créent un abonnement associé à un sujet pour recevoir des messages de celui-ci. La communication peut être de type un à plusieurs, comme le montre le schéma suivant.

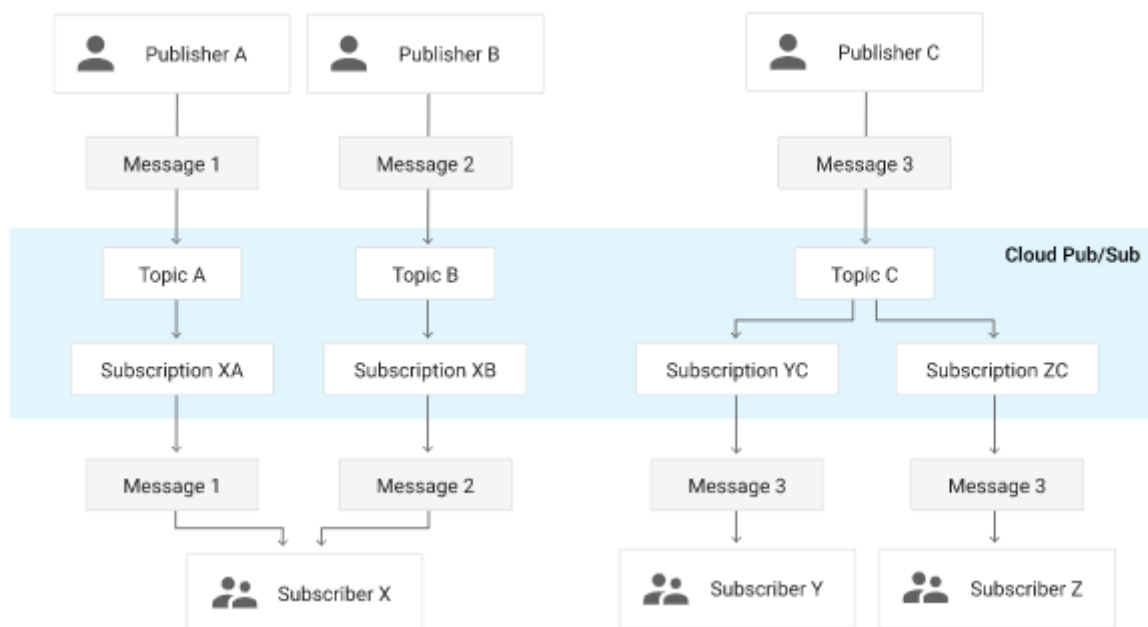


Figure 29 : Relations éditeur-abonné

— Cas d'utilisation courants

On peut utiliser Pub/Sub dans un workflow asynchrone. Par exemple, une application de traitement de commandes peut passer une commande dans un sujet, à partir de laquelle elle

peut être traitée par un ou plusieurs nœuds. En plus, on peut l'utiliser dans une distribution de notifications d'évènements. Par exemple, un service qui accepte les inscriptions d'utilisateurs peut envoyer des notifications chaque fois qu'un nouvel utilisateur s'inscrit, et les services en aval peuvent s'abonner pour recevoir des notifications de l'évènement.

5.7.2.4 Firebase Realtime Database

La base de données Firebase est une base de données hébergée dans le Cloud. Les données sont stockées au format JSON et synchronisées en temps réel avec chaque client connecté. Lorsque nous créons des applications multiplateformes, tous nos clients partagent une instance de base de données en temps réel et reçoivent automatiquement des mises à jour avec les données les plus récentes.

La base de données Firebase est une base de données NoSQL et, en tant que telle, a des optimisations et des fonctionnalités différentes par rapport à une base de données relationnelle. L'API de base de données Firebase est conçue pour n'autoriser que les opérations qui peuvent être exécutées rapidement. Cela nous permet de créer une excellente expérience en temps réel qui peut servir des millions d'utilisateurs sans compromettre la réactivité.

5.7.3 Déploiement de la solution

5.7.3.1 Déploiement des fonctions

Pour effectuer des déploiements, nous devons transférer une archive contenant le code source de notre fonction vers un bucket Google Cloud Storage. Une fois le code source importé, Cloud Build compile automatiquement notre code dans une image de conteneur, puis transfère cette image à Container Registry. Cloud Functions utilise cette image pour créer le conteneur qui exécute notre fonction.

Nous pouvons déployer des fonctions Cloud Functions depuis notre ordinateur local, depuis notre dépôt source GitHub ou directement depuis l'API Cloud Functions.

Lors de déploiement, Cloud Functions recherche des fichiers spécifiques, en fonction de notre environnement d'exécution. Avant que le déploiement de notre fonction ne soit finalisé, Cloud Functions enverra une requête de test pour confirmer notre déploiement.

Tout d'abord, on doit activer l'API Cloud Fonctions pour déployer une fonction. Ce déploiement nécessite deux fichiers qui vont être rédigés à l'aide de l'éditeur intégré. Le premier fichier nommé `package.json` qui contient les bibliothèques à utiliser et la deuxième nommée

index.js qui contient la fonction à exécuter. Dans notre cas, on a choisi ces configurations pendant le déploiement :

- Mémoire allouée : 256 Mo pour notre traitement est suffisant,
- Déclencheur : Cloud Storage pour la fonction d'extraction du texte par contre Cloud Pub/Sub pour les autres fonctions,
- Dans le cas où le déclencheur Cloud Storage, il faut indiquer le nom de bucket à surveiller et dans le cas des autres fonctions, il faut indiquer le nom du sujet Cloud Pub/Sub,
- L'environnement d'exécution du code (dans notre cas c'est Nodejs8),
- La fonction à exécuter, c'est le nom de la fonction à exécuter dans le fichier index.js. Enfin, on doit cliquer sur déployer.

Les figures ci-dessous représentent les fenêtres de déploiement de nos fonctions Cloud.

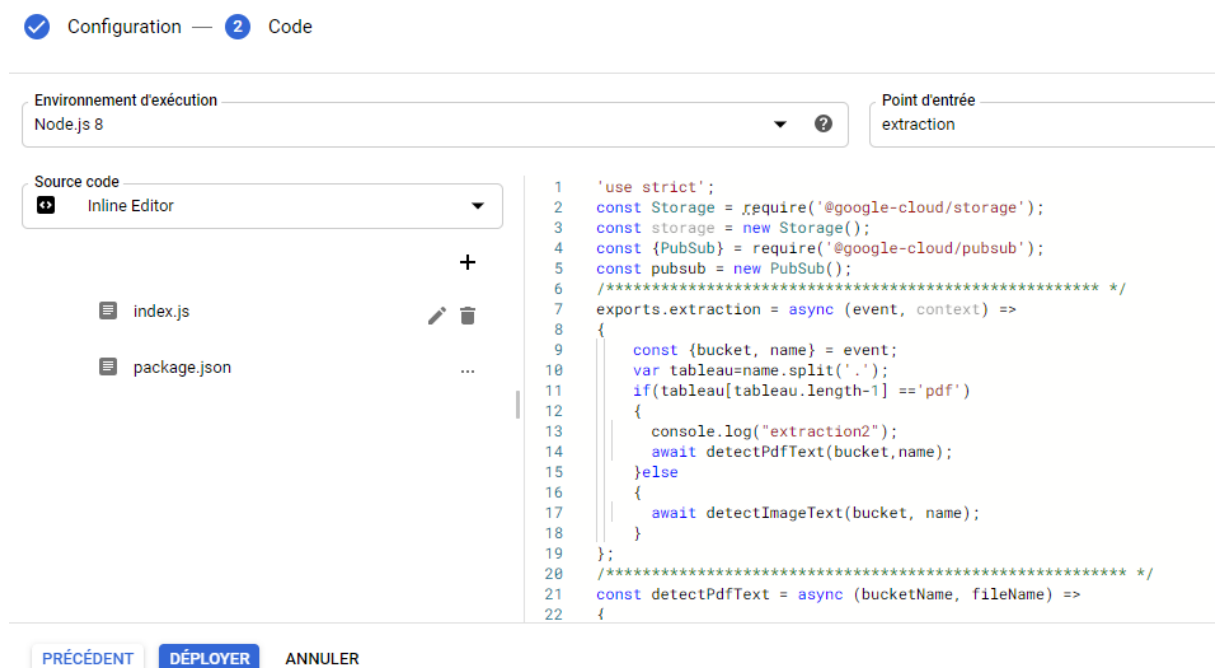


Figure 30 : Fenêtre de déploiement de la fonction extraction

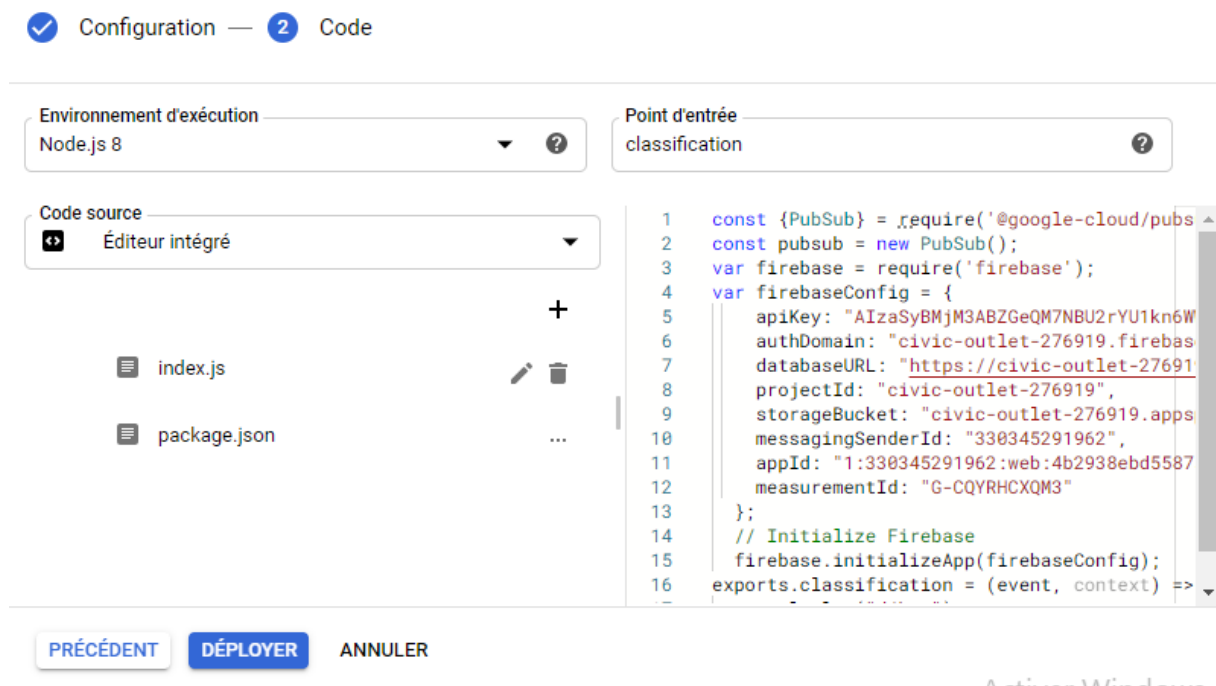


Figure 31 : Fenêtre de déploiement de la fonction classification

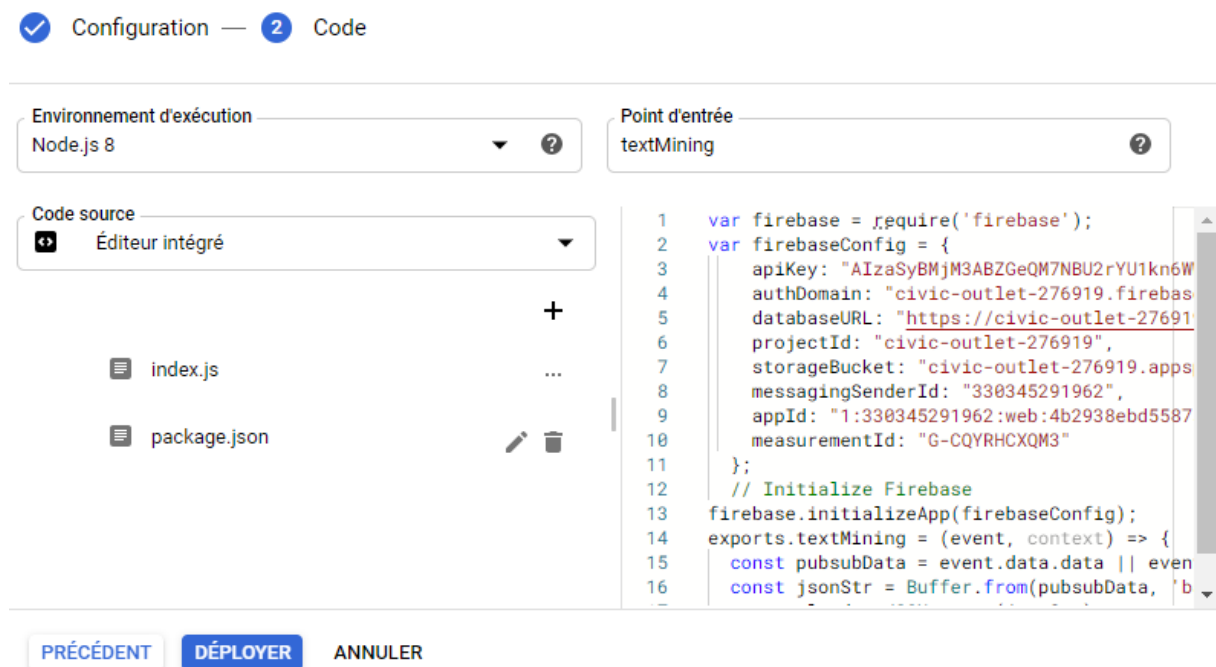


Figure 32 : Fenêtre de déploiement de la fonction textMining

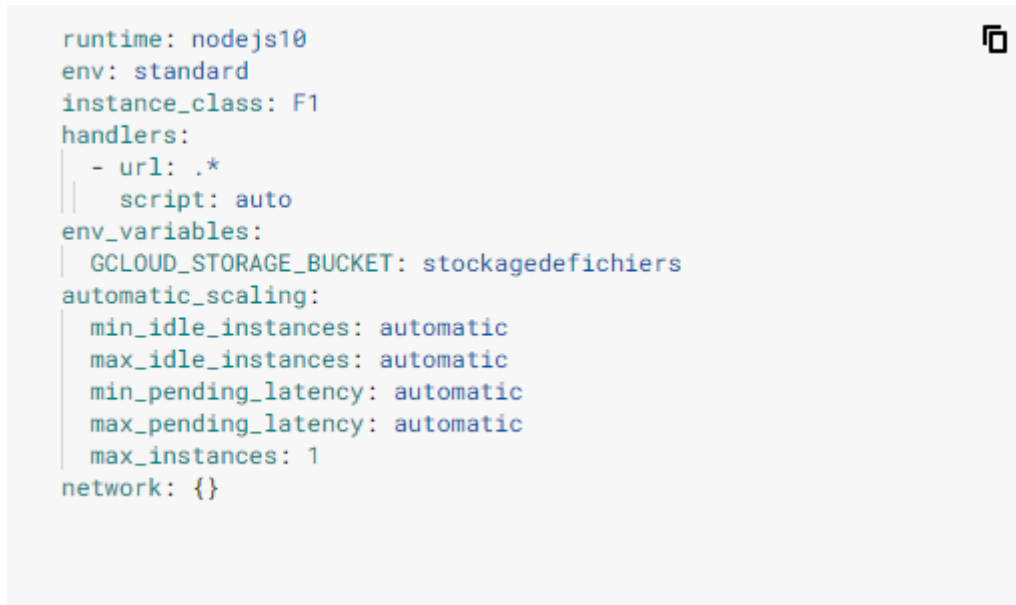
5.7.3.2 Déploiement de service web

Nous avons utilisé l'outil gcloud pour déployer notre service web Node.js sur App Engine. Tout d'abord, on doit installer le SDK Cloud afin de pouvoir exploiter l'outil de ligne de commande gcloud. En plus, on doit ajouter un fichier de configuration de déploiement

app.yaml. Ensuite, dans le dossier où se trouve ce fichier de configuration, on exécute la commande suivante dans le terminal : `gcloud app deploy`.

Les modules Node.js sont installés dans le cloud tels qu'ils sont répertoriés dans le fichier `package.json` et notre service est lancé en utilisant `npm start`.

La figure 33 représente la configuration de notre service web.



```
runtime: nodejs10
env: standard
instance_class: F1
handlers:
- url: .*
  script: auto
env_variables:
  GCLOUD_STORAGE_BUCKET: stockagedefichiers
automatic_scaling:
  min_idle_instances: automatic
  max_idle_instances: automatic
  min_pending_latency: automatic
  max_pending_latency: automatic
  max_instances: 1
network: {}
```

Figure 33 : La configuration de notre service web

Comme indiqué dans la figure ci-dessus, nous avons choisi de déployer notre service web sur un environnement standard de produit App Engine. L'environnement standard App Engine est basé sur des instances de conteneur qui s'exécutent sur l'infrastructure de Google. L'environnement standard App Engine permet de concevoir et de déployer des applications qui s'exécutent de manière fiable, même lorsqu'elles doivent faire face à des charges importantes et gérer de grandes quantités de données.

Les applications s'exécutent dans un environnement de bac à sable sécurisé, ce qui permet à l'environnement standard App Engine de répartir les requêtes sur plusieurs serveurs et d'adapter ces derniers à la demande de trafic. Celui-ci est indépendant du matériel, du système d'exploitation et de l'emplacement physique du serveur.

Concernant les tarifs, un quota gratuit pour l'utilisation des ressources App Engine s'applique aux applications exécutées dans l'environnement standard. Au-delà de cette limite, l'utilisation des ressources App Engine est facturée comme décrit la figure suivante :

Classe d'instance	Coût à l'heure par instance
B1	0,05 \$
B2	0,10 \$
B4	0,20 \$
B4_1G	0,30 \$
B8	0,40 \$
F1	0,05 \$
F2	0,10 \$
F4	0,20 \$

Figure 34 : Tarifs de produit App Engine

Les instances-heure sont comptabilisées entre le démarrage et l'arrêt de l'instance, et en fonction du type de scaling défini. Les règles suivantes sont appliquées :

- Scaling de base ou automatique : Le décompte se termine 15 minutes après que l'instance a fini de traiter sa dernière requête,
- Scaling manuel : Le décompte se termine 15 minutes après l'arrêt de l'instance.

L'utilisation des instances est facturée selon leur temps d'activité, à un taux horaire donné. Des quotas gratuits sont appliqués spécifiquement aux classes d'instances «F» et «B».

Ressource	Quota gratuit
Instances "F"	28 heures gratuites par jour
Instances "B"	9 heures gratuites par jour

Figure 35 : Les quotas gratuits de produit App Engine

5.8 Choix technique

Dans cette sous-partie, nous définissons l'environnement d'exécution choisi pour l'implémentation : Node.js.

Node.js est une plateforme logicielle libre en JavaScript intégrant un serveur HTTP. Son fonctionnement est basé sur une boucle événementielle lui permettant de supporter de fortes montées en charge.

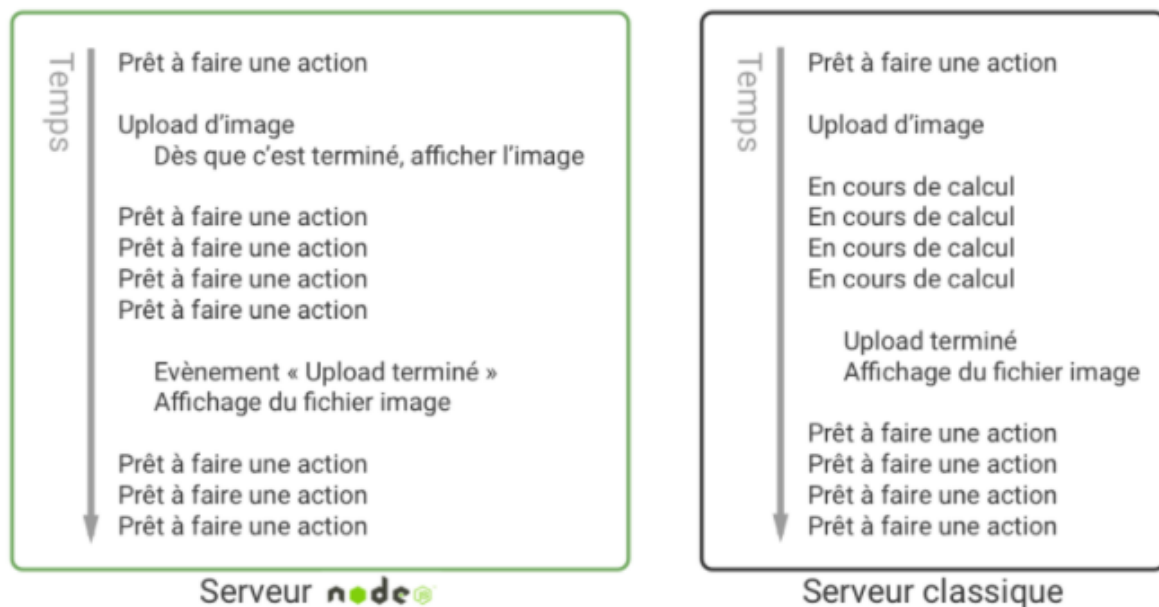


Figure 36 : Serveur Node.js VS Serveur classique

L'utilisation de Node.js en tant que serveur web permet de traiter un gros volume de requêtes simultanément de manière efficace. Cette performance élevée s'explique par une conception asynchrone (modèle non bloquant) permettant d'éviter les attentes. Ainsi, plusieurs requêtes peuvent être lancées en parallèle. Les résultats sont traités au fil de l'eau. Node utilise le compilateur JavaScript V8 de Google focalisé sur les performances et la sécurité. Ainsi, les progrès de V8 impactent directement Node.js.

— Les développeurs Node aiment le partage

Node.js rassemble une grande communauté de développeurs passionnés. Il existe des Mooc, des webinaires ou des cours en ligne pour apprendre à utiliser Node.js. Les articles spécialisés et les événements permettent de partager les avancées de Node.js. Des Meetup, des conférences ou des formations sont accessibles.

— **Le gestionnaire de packages npm**

Initialement gestionnaire de packages de Node.js, npm est aujourd'hui le gestionnaire de packages du monde JavaScript. On y trouve aussi bien des modules pour le backend que pour le frontend. Les développeurs sont très actifs. Ils contribuent constamment à l'amélioration des bibliothèques Node.js. Npm compte aujourd'hui plus de 500 000 packages et le nombre ne cesse d'augmenter, bien plus rapidement que pour les autres langages.

— **Une technologie stable et éprouvée**

Node.js n'est plus la petite dernière des technologies. C'est une technologie utilisée et éprouvée par les géants du web : Netflix, Trello, PayPal, LinkedIn, Walmart, Uber, Medium, Groupon, Ebay ou la NASA. Une politique très stricte a été mise en place au sujet des mises à jour et du suivi des versions Node.js. Les deux points importants sont les suivants :

- Chaque mise à jour est suivie d'une période de 18 mois de support actif + 12 mois de maintenance,
- Il n'y aura pas de gros «Breaking Changes» avec Node.js. Les versions successives ne casseront pas notre application et on ne nous demandera pas non plus d'apprendre TypeScript pour la version X. On peut dire que les nouvelles versions sont Smooth...

Node.js offre des performances optimales, une communauté super active, des outils à la pointe et une stabilité assurée.

5.9 Aperçu des résultats obtenus

Dans cette section, on met en disposition quelques captures d'écran des résultats du travail réalisé. La première figure représente un justificatif de note de frais d'un collaborateur chez la société Redlean. Ce justificatif représente une dépense professionnelle et doit être remboursés. Pour faciliter le déroulement du processus de gestion des notes de frais, notre collaborateur va utiliser notre solution intelligente pour générer automatiquement et d'une manière autonome la date, le montant TTC et la classe de sa note de frais.

La figure 37 représente le résultat obtenu.

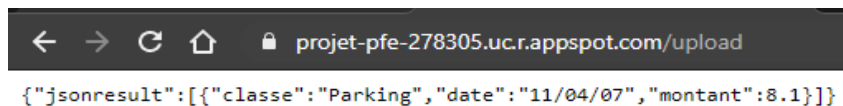


Figure 37 : Résultat du notre solution

En plus, nous avons testé la solution avec un autre justificatif de note de frais et la figure 38 représente le résultat obtenu.

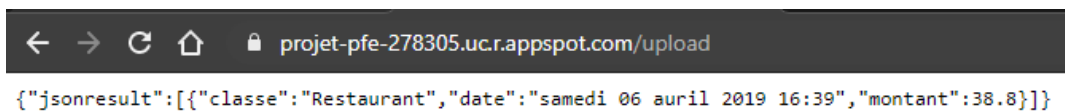


Figure 38 : Résultat du notre solution

Enfin, d'après notre diagramme des cas d'utilisation, l'administrateur doit configurer la solution pour assurer le bon déroulement des algorithmes. Il doit saisir les mots clés qui décrivent les différentes classes de notes de frais selon les besoins de la société Redlean et la figure 39 représente l'interface web qui lui permet d'ajouter d'une manière dynamique les mots clés à notre base de données.

interface de configuration

[vider la page](#)

restaurant :

péage :

parking :

train :

hotel :

avion :

Activer Windows
Accédez aux paramètres pour activer W

Figure 39 : Interface de configuration

Conclusion

Tout au long de ce chapitre, nous avons bien détaillé les étapes de processus de traitement de note de frais, ainsi nous avons présenté notre choix technologique et quelques aperçus des résultats obtenues.

Conclusion et perspectives

Le présent rapport indique les différentes étapes pour lesquelles nous sommes passés pour mettre en place un système de génération automatique des dépenses permettant à un collaborateur de gérer des notes de frais d'une manière intelligente et autonome. Ainsi, ce système permet à l'administrateur de configurer la solution selon les besoins de la société Redlean. Nous avons réussi à digitaliser une partie de processus de gestion des notes de frais en supprimant la saisie fastidieuse et répétitive.

Plusieurs technologies ont été exploitées durant la période de stage, afin d'atteindre le résultat souhaité telles que Node.js, Google Cloud Functions, Google App engine, Google Cloud Storage, Google Pub/Sub, etc... .

Le stage a constitué une expérience très riche sur le plan personnel et technique puisqu'il a permis une véritable porte d'entrée motivante et intéressante au monde professionnel. Ainsi, le stage m'a permis de mettre en œuvre et de consolider les connaissances acquises tout au long de mon cursus académique. Il a de même permis d'améliorer mes connaissances dans les architectures logicielles.

En termes de perspectives, on peut continuer à digitaliser le processus de gestion des notes de frais. On peut enrichir notre application par un module qui permet au service comptable de récupérer automatiquement les dépenses dans le logiciel de comptabilité sans aucune ressaisie. On peut aussi ajouter un module d'analyse et reporting qui permet d'analyser les dépenses par catégorie et période grâce à des tableaux de bord et des statistiques.

Bibliographie et Webographie

- [W1] <https://redlean.io/fr/> : Le site officiel de la société Redlean
- [W2] <https://www.linkedin.com/company/redlean/> : Page LinkedIn de la société Redlean
- [W3] <https://openclassrooms.com/fr/courses/4511226-gerez-votre-projet-avec-une-equipe-scrum> : La gestion de projet agile
- [W4] <https://www.sitepoint.com/developers-rest-api/> : Qu'est-ce qu'une API Rest ?
- [W5] <https://blog.ippon.fr/2017/06/09/les-architectures-serverless/> : Les architectures Serverless
- [W6] <https://www.cloudflare.com/fr-fr/learning/serverless/glossary/serverless-microservice/> : Explication des microservices sans serveur
- [W7] <https://cloud.google.com/> : Le site officiel de Google Cloud Platform
- [W8] <https://www.expensya.com/fr/> : Le site officiel d'Expensya
- [W9] <https://www.n2f.com/note-de-frais> : Le site officiel de N2F
- [W10] <https://www.rydoo.com/fr/> : Le site officiel de Rydoo Expense
- [Bib1] Pascal Roques. UML 2 par la pratique. Editions Eyrolles, août 2006
- [W11] <https://swagger.io/docs/specification/basic-structure/> : La structure basique d'un document OpenAPI
- [W12] <https://cloud.google.com/vision/docs/ocr> : Détecter le texte dans les images
- [W13] <https://cloud.google.com/vision/docs/pdf> : Détecter le texte dans les fichiers (PDF/TIF)
- [W14] <https://cloud.google.com/functions> : Google Cloud Functions
- [W15] <https://cloud.google.com/appengine> : Google App Engine
- [W16] https://cloud.google.com/functions/docs/tutorials/ocr?hl=fr#deploying_the_functions : Tutoriel sur la reconnaissance optique des caractères (OCR)

[W17] <https://cloud.google.com/appengine/docs/standard/nodejs/building-app/deploying-web-service?hl=fr> : Déployer votre service web Node.js

[W18] <https://nodejs.org/en/> : Le site officiel de Node.js

[W19] https://www.w3schools.com/nodejs/nodejs_intro.asp : Qu'est-ce que Node.js?

Résumé

Titre : Création d'une solution intelligente pour l'automatisation de génération des dépenses.

Le présent travail réalisé au sein de Redlean services s'inscrit dans le cadre du projet de fin d'études en vue d'obtenir le diplôme National d'Ingénieur en Informatique. Le projet consiste à mettre en place un système de génération automatique des dépenses. Le but de ce système est de digitaliser le processus de gestion des notes de frais.

Mot clés : Note de frais, Google Cloud Platform, Cloud, Node.js, Google Cloud Functions, Google App Engine, Microservices.

Abstract

Title: Creating a smart solution for automating expense generation.

The present work carried out within Redlean services is part of the end-of-studies project to obtain the National Diploma of Computer Engineering. The project consists of setting up an automatic generation system for expenses. The aim of this system is to digitize the expense report management process.

Keywords : Expense report, Google Cloud Platform, Cloud, Node.js, Google Cloud Functions, Google App Engine, Microservices.

