

UNIVERSITE DE YAOUNDE I

-----  
ECOLE NATIONALE  
SUPERIEURE  
POLYTECHNIQUE  
-----

DEPARTEMENT DU GENIE  
INFORMATIQUE



UNIVERSITY OF YAOUNDE I

-----  
NATIONAL ADVANCED  
SCHOOL OF ENGINEERING  
-----

COMPUTER SCIENCE  
DEPARTMENT

DÉPARTEMENT DU GÉNIE INFORMATIQUE  
ANNÉE ACADÉMIQUE : 2019 - 2020

RAPPORT DE STAGE

---

## CONCEPTION D'UNE API REST DE GESTION DES ÉCHÉANCIERS ET DES RELANCES

---



*Auteur:*

MENGONG MEYANGA SIMON  
PIERRE

*Encadreur:*

PR. THOMAS DJOTIO NDIÉ

NOVEMBRE 2020

## **REMERCIEMENTS**

Je remercie tout d'abord le SEIGNEUR pour m'avoir accordé vie et santé tout au long de mon stage.

Je remercie également le Professeur THOMAS DJOTIO qui a énormément contribué dans la rédaction de ce mémoire, et aussi pour l'opportunité qu'il m'a donnée de faire le stage dans le laboratoire du CETIC et ensuite pour ses nombreux conseils.

Je tiens aussi à remercier ma FAMILLE qui m'a beaucoup soutenu durant mon cursus scolaire financièrement, matériellement et par ses constantes prières.

## ABRÉVIATIONS

Acronymes	Significations
<b>API</b>	Application Programming Interface
<b>CETIC</b>	Centre Africain d'Excellence en Technologies de l'Information
<b>HTTP</b>	HyperText Transfer Protocole
<b>JSON</b>	JavaScript Object Notation
<b>ORM</b>	Object-Relational Mapping
<b>REST</b>	REpresentational State Transfer
<b>URL</b>	Uniform [Universal] resource locator

## RÉSUMÉ

**N**ous remarquons aujourd’hui une recrudescence des interactions interpersonnelles basées tant sur les communications entre individus que sur les échanges de biens et de services entre ceux-ci. Dans l’optique de fluidifier ces échanges, des solutions de vente de produit en ligne, dites « E-commerce » ont vu le jour. Ces solutions sont sans doute multiples mais généralement peu adaptées au contexte qui est le nôtre. Il est en général demandé au client de régler sa facture en totalité et en une fois. Ce qui pose un problème aux clients n’ayant pas à disposition la somme demandée. C’est pourquoi dans ce document nous présenterons une solution permettant de régler le problème de paiement des factures en offrant la possibilité aux clients de pouvoir payer leurs factures en plusieurs étapes. Pour la mise sur pied de notre solution nous avons opté après mures analyses pour les meilleurs outils. C’est ainsi que nous sommes parvenus à mettre sur pied une solution permettant d’établir des échéanciers, et d’effectuer des relances, et tous ceci sous la forme d’une API, ce qui facilitera son intégration à des sites déjà existants.

**Mots-clés : API, E-commerce**

# TABLE DES MATIÈRES

	Page No
<b>Abréviations</b>	<b>ii</b>
<b>Liste des tableaux</b>	<b>vii</b>
<b>Table des figures</b>	<b>viii</b>
<b>INTRODUCTION GÉNÉRALE</b>	<b>ix</b>
Contexte . . . . .	x
<b>1 Présentation du laboratoire CETIC</b>	<b>1</b>
1.1 Vue d'ensemble . . . . .	2
1.2 Vision . . . . .	2
1.3 Mission . . . . .	2
1.4 Stratégie . . . . .	2
1.5 Organigramme . . . . .	3
<b>2 ÉTAT DE L'ART</b>	<b>5</b>
2.1 Généralités sur les APIs . . . . .	6
2.1.1 Les caractéristiques des APIs . . . . .	6
2.1.2 Fonctionnement . . . . .	7
2.1.3 Structure fonctionnelle d'une API . . . . .	8
2.1.4 Les avantages à utiliser des APIs . . . . .	8
2.1.5 Exemples d'API . . . . .	9
2.2 Présentation des solutions existantes . . . . .	10
2.2.1 Contexte . . . . .	10
2.2.2 Solutions de relance . . . . .	10
2.2.3 Solutions d'échéancier . . . . .	11
2.3 Problématique . . . . .	11

2.4	Solutions apportées . . . . .	12
2.5	Conclusion . . . . .	12
<b>3</b>	<b>ANALYSE ET CONCEPTION</b>	<b>13</b>
3.1	Analyse . . . . .	15
3.1.1	Définition des termes clés . . . . .	15
3.1.2	Cahier de charge . . . . .	15
3.1.3	Les acteurs . . . . .	16
3.1.4	Les cas d'utilisations . . . . .	17
3.1.5	Description des cas d'utilisations . . . . .	17
3.2	Description textuelle des cas d'utilisations principaux . . . . .	18
3.2.1	Définir un échéancier . . . . .	18
3.2.2	Enregistrer un paiement . . . . .	18
3.2.3	Enregistrer une relance . . . . .	19
3.3	Cas d'utilisations internes au système . . . . .	19
3.3.1	Contrôler les échéanciers . . . . .	19
3.3.2	Faire une relance . . . . .	20
3.4	Conception . . . . .	21
3.4.1	Diagramme de cas d'utilisations . . . . .	21
3.4.2	Diagramme de séquence . . . . .	22
3.4.3	Diagramme de classe . . . . .	23
3.5	Conclusion . . . . .	23
<b>4</b>	<b>IMPLEMENTATION ET RESULTAT FINAL</b>	<b>24</b>
4.1	Les outils utilisés pour la création de notre système . . . . .	25
4.1.1	Python . . . . .	25
4.1.2	Django . . . . .	25
4.1.3	Django Rest Framework . . . . .	25
4.1.4	ReactJS . . . . .	26
4.2	Quelques vues de la réalisation . . . . .	27
4.2.1	Page d'authentification . . . . .	27
4.2.2	Page d'accueil . . . . .	27
4.2.3	Page Nouveau Contrat . . . . .	28
4.2.4	Page Client . . . . .	30
4.3	conclusion . . . . .	30

---

<b>Conclusion</b>	<b>31</b>
<b>Bibliographie</b>	<b>32</b>

## **LISTE DES TABLEAUX**

**TABLE**

**Page No**



## TABLE DES FIGURES

FIGURE	Page No
1.1 Organigramme du CETIC . . . . .	3
2.1 Illustration du fonctionnement d'une API . . . . .	8
3.1 Diagramme de contexte . . . . .	16
3.2 Diagramme de cas d'utilisation . . . . .	21
3.3 diagramme de séquence (enregistrer échéancier) . . . . .	22
3.4 diagramme de classe . . . . .	23
4.1 Page d'authentification . . . . .	27
4.2 Page d'accueil . . . . .	28
4.3 Page Nouveau Contrat . . . . .	29
4.4 Page Client . . . . .	30

# INTRODUCTION GÉNÉRALE

A l'Ecole Nationale Supérieure Polytechnique de l'Université de Yaoundé I (ENSP-UY1), dans son cursus d'ingénieur, il est obligatoire pour un étudiant de 4ème année d'effectuer deux mois de stage en entreprise ou en laboratoire dans le but de renforcer ses capacités de production. C'est ce qui me conduit dans le cadre de mon stage au sein l'équipe NetSAP du laboratoire du CETIC (hébergée à l'ENSP).

Un chercheur dynamique a mis sur pied une équipe qui travaille sur des idées innovatrices centrées sur la mise à disposition des populations du Cameroun en particulier et du monde en générale des produits et services proposés par des fournisseurs au moyen d'applications Web et mobiles utilisant le réseau internet. Après une étude et une analyse approfondis des besoins il a mis sur pied un système d'informations qui sera exploité par ces diverses applications. C'est alors que j'ai été chargé de contribuer à :

La conception d'une API REST de gestion des Échéanciers et des Relances

De la stratégie de résolution des problèmes que nous avons suivi pour aboutir à cette tâche, au travail lui-même et à la présentation des résultats, nous avons mis en pratique diverses connaissances extraites des cours suivis depuis le début de notre formation dans cette prestigieuse école. Associer à cela, la qualité de nos échanges avec les autres membres de l'équipe a également joué un rôle important dans la concrétisation de ce projet.

## Contexte

Dans l'environnement actuel il est devenu possible et même très facile de réaliser des achats sur des sites de vente en ligne et de régler à travers divers moyens de paiement ses commandes. Mais nous sommes amenés à constater qu'il n'y a pas dans la grande majorité des cas de solution permettant à des personnes ne disposant pas de la totalité de la somme demandée, de rentrer en possession du produit souhaité ce qui ne contribue pas à faire augmenter le rendement de ces sites. C'est pourquoi il a été observé de mettre sur pied une solution de paiement de facture en plusieurs étapes reposant sur la définition d'échéanciers, et dans le soucis de permettre aux sites Web déjà en place de pouvoir y avoir accès sans avoir à reconstruire tout le site, la solution va être implémenter sous la forme d'un « plugin » lui permettant ainsi une intégration plus facile.

C'est dans cette optique que nous allons dans ce projet réaliser une API Web qui va permettre :

- la définition d'échéanciers de paiement qui sera gérer de manière automatique par celle-ci ; et
- L'envoi des messages d'alerte aux clients pour les prévenir de l'état des échéanciers, des paiements et des prochaines échéances.

# CHAPITRE 1

## PRÉSENTATION DU LABORATOIRE CETIC

Dans ce chapitre, nous allons examiner le laboratoire dans lequel nous avons effectué notre stage. Nous allons donner une vue d'ensemble, ses visions, ses missions, sa stratégie, entre autres.

### Contents

1.1	Vue d'ensemble . . . . .	2
1.2	Vision . . . . .	2
1.3	Mission . . . . .	2
1.4	Stratégie . . . . .	2
1.5	Organigramme . . . . .	3

## 1.1 Vue d'ensemble

Le Centre africain d'Excellence en Technologies de l'Information et de la Communication (CETIC) est un centre de formation et de recherche créé au sein de l'Ecole Nationale Supérieure Polytechnique de l'Université de Yaoundé I par le gouvernement du Cameroun avec l'assistance de la Banque mondiale. Le CETIC dispose d'un laboratoire avec de nombreuses équipes de recherche travaillant dans divers domaines. L'équipe Network Services Applications (NetSAP) est l'une d'elle qui a l'intention d'explorer les questions de recherche qui est liée à La mise en réseau d'ordinateurs.

## 1.2 Vision

- **La vision du CETIC** est de construire une Afrique où le développement des Technologies de l'Information et de la Communication (TIC) est inter-connecté avec le développement durable des défis dans l'intérêt de tous.
- **La vision de NetSAP** est de démocratiser la connectivité des réseaux sans fil, les services et applications dans les pays en développement.

## 1.3 Mission

- **La mission du CETIC** est de développer des programmes novateurs de formation et de recherche pour satisfaire le marché en matière d'expertise en TIC et soutenir la production des services électroniques à forte valeur ajoutée et fortement compétitifs qui contribuent à l'émergence de l'Afrique.
- **La mission du NetSAP** est d'exploiter le potentiel des personnes via le contenu du réseau et service : éducation ; santé, transports, ...

## 1.4 Stratégie

- Le CETIC avait mis en place un ensemble de programmes d'éducation bien connus et réclamant maîtrise et doctorat / doctorat depuis 2014 avec trois options : génie logiciel, Gestion des systèmes d'information, des réseaux et des télécommunications.
- Le CETIC a établi des partenats avec plus de 12 universités et grandes écoles, notamment : EPFL, Buéa, Nsukka, Cheikh Anta Diop, etc.

- Le CETIC a établi des partenariats avec plus de 5 instituts de recherche, notamment : NASEY, IRD, Lirima, Ummisco, etc.
- Le CETIC a également des partenariats fructueux avec plus de 5 sociétés : CAMTEL, ANTIC, CCIM, ITG-Store, etc.

## 1.5 Organigramme

Pour exécuter cette stratégie, voici comment l'équipe dirigeante est organisée hiérarchiquement :

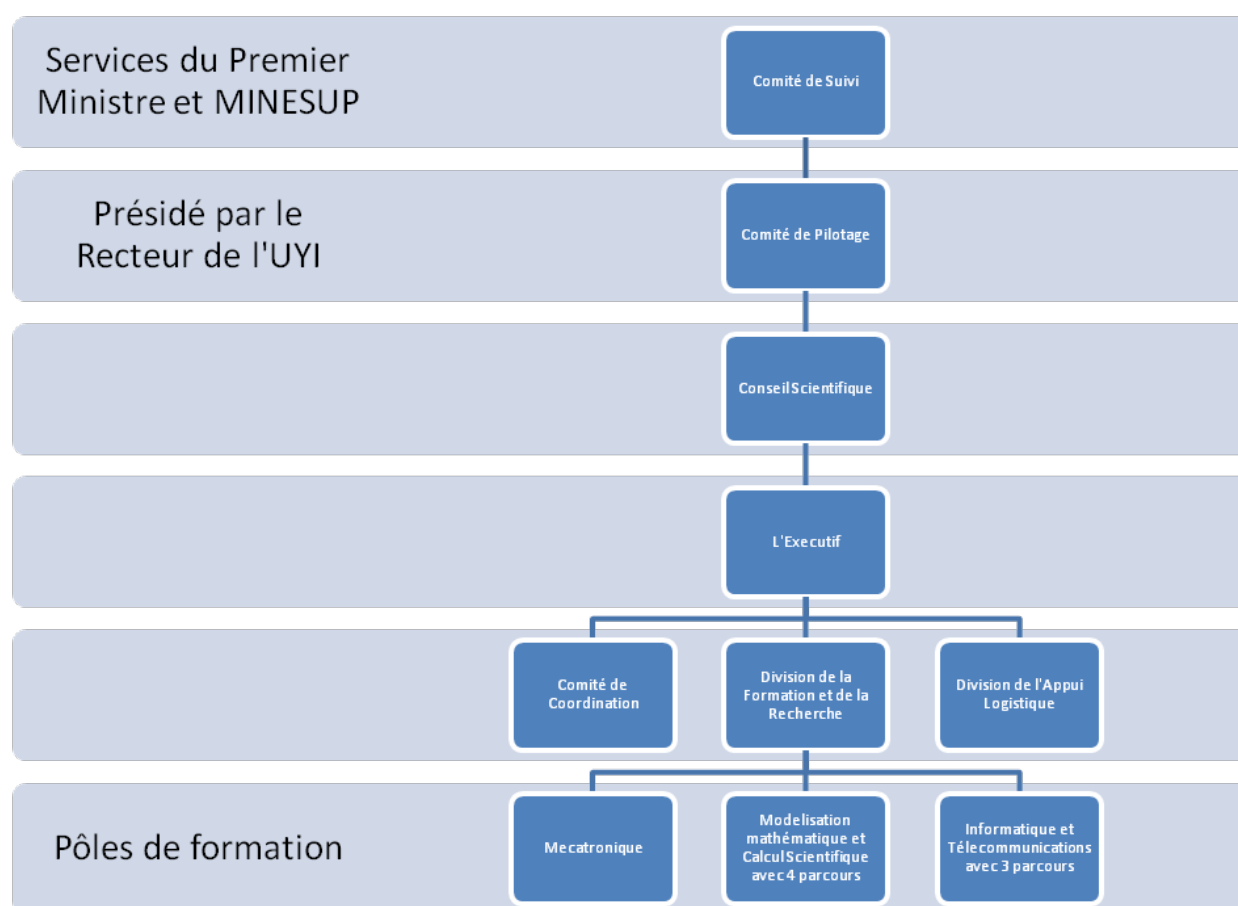


FIGURE 1.1. Organigramme du CETIC

Sur cette figure, vous pouvez voir qu'au premier niveau de la supervision on a un Comité de Suivi du MINESUP suivi du Comité de Pilotage organisé à l'Université de Yaoundé I (UYI). Et tout en bas, on a des pôles de formation dont le rôle est d'organiser et dispenser

les cours. En plus de ce dernier, on a la Division de la formation et des recherches qui coordonnent les activités de formation et de recherche du CETIC.

## ÉTAT DE L'ART

Dans ce chapitre, après avoir brièvement présenté les concepts de bases relatifs à notre sujet, nous nous intéresserons aux solutions existantes et aux limites de celles-ci, par la suite nous soulèverons le problème à résoudre et nous présenterons la solution.

### Contents

2.1	Généralités sur les APIs . . . . .	<b>6</b>
2.1.1	Les caractéristiques des APIs . . . . .	6
2.1.2	Fonctionnement . . . . .	7
2.1.3	Structure fonctionnelle d'une API . . . . .	8
2.1.4	Les avantages à utiliser des APIs . . . . .	8
2.1.5	Exemples d'API . . . . .	9
2.2	Présentation des solutions existantes . . . . .	<b>10</b>
2.2.1	Contexte . . . . .	10
2.2.2	Solutions de relance . . . . .	10
2.2.3	Solutions d'échéancier . . . . .	11
2.3	Problématique . . . . .	<b>11</b>
2.4	Solutions apportées . . . . .	<b>12</b>
2.5	Conclusion . . . . .	<b>12</b>



## 2.1 Généralités sur les APIs

« API » est un acronyme anglo-saxon signifiant Application Programming Interface. Une API est une interface, un contrat passé entre deux systèmes informatiques pour leur permettre de communiquer. Une API, c'est ce qui permet à deux systèmes informatiques totalement indépendants de se parler de façon automatique. Plus précisément, une API est le mode d'emploi qui permet à un système informatique de faire appel à des fonctionnalités d'un autre système informatique : elle permet donc de les rendre interopérables entre eux.

- **Application** : c'est un service accessible par un humain ou un programme informatique, directement utilisé pour réaliser une tâche, ou un ensemble de tâches élémentaires d'un même domaine ou formant un tout. **Exemple** : un portail open data, un réseau social, une brosse à dent connectée.
- **Programmation** : Un programme est un ensemble de fonctions informatiques écrites par un développeur (ou une intelligence artificielle) qui exécute des tâches à sa place.
- **Interface** : c'est la porte d'entrée à travers laquelle le programme pourra interagir avec l'application. Une interface définit la frontière de communication entre deux entités, comme des éléments de logiciel, des composants de matériel informatique, ou des utilisateurs. Elle se réfère généralement à une image abstraite qu'une entité fournit d'elle-même à l'extérieur.

Ces interfaces de programmation permettent d'enrichir un programme avec des fonctions issues d'un autre logiciel pour développer des fonctionnalités plus poussées ou importer des données pré-organisées, traitées et/ou intégrées ailleurs.

Pour cela, deux possibilités :

- Développer ses propres interfaces pour un usage interne ;
- Ou utiliser des API conçues et publiées par d'autres organismes.

### 2.1.1 Les caractéristiques des APIs

#### 2.1.1.1 Dépendance au langage

Une API peut être utilisable dans un unique langage de programmation ou être indépendante des langages. Dans le second cas un langage intermédiaire comme XML peut être utilisé comme format de données pour les requêtes aux fonctions et méthodes.

#### **2.1.1.2 Licence d'utilisation**

Elle est sous licence libre et utilisable sans frais par tout programmeur, ou sous licence propriétaire et accessible uniquement à une communauté restreinte, ce qui est le cas par exemple des API de consoles de jeux.

#### **2.1.1.3 Niveau de langage**

On distingue d'une part l'API de haut niveau, en matière de langage de programmation, comme les API graphiques et d'autre part l'ABI (Application Binary Interface), proche du système, comme la Linux Standard Base ou les interfaces de pilotes de matériels.

### **2.1.2 Fonctionnement**

La création et l'utilisation des interfaces de programmation est un sujet incontournable de la programmation contemporaine. Une application se sert généralement de nombreuses interfaces de programmation ; mises en oeuvre par des bibliothèques logicielles ou des services web qui peuvent eux-mêmes se servir d'autres interfaces de programmation.

En architecture orientée services les applications peuvent dépendre de fonctionnalités tierces offertes par des logiciels via des interfaces de programmation mises en oeuvre par des services web.

Les interfaces de programmation permettent de gagner du temps par la collaboration et la spécialisation des équipes de développement de logiciel. Par exemple aujourd'hui plus personne n'écrit un SGBD maison pour une application informatique. Les programmeurs réutilisent les SGBD existant dans le commerce, fournis par des entreprises spécialisées dans ce type de produit, et se concentrent sur la logique propre à leur application. De nombreux produits d'infrastructure sont ainsi disponibles sous forme de Framework ou de bibliothèque.

Une interface de programmation permet par exemple à un programme d'accéder aux services offerts par le système d'exploitation qui héberge le programme. L'interface sockets est un exemple classique d'interface de programmation qui permet à un programme d'exploiter les possibilités de la couche réseau du système d'exploitation.

### 2.1.3 Structure fonctionnelle d'une API

Pour résumer, on peut dire qu'une API est ce qui permet à un programme informatique de profiter des fonctionnalités d'un autre programme informatique, tout autant que votre écran (associé à une souris et un clavier s'il n'est pas tactile) est ce qui vous permet de profiter des fonctionnalités d'un programme informatique. On peut représenter le concept d'API par la figure 2.1.

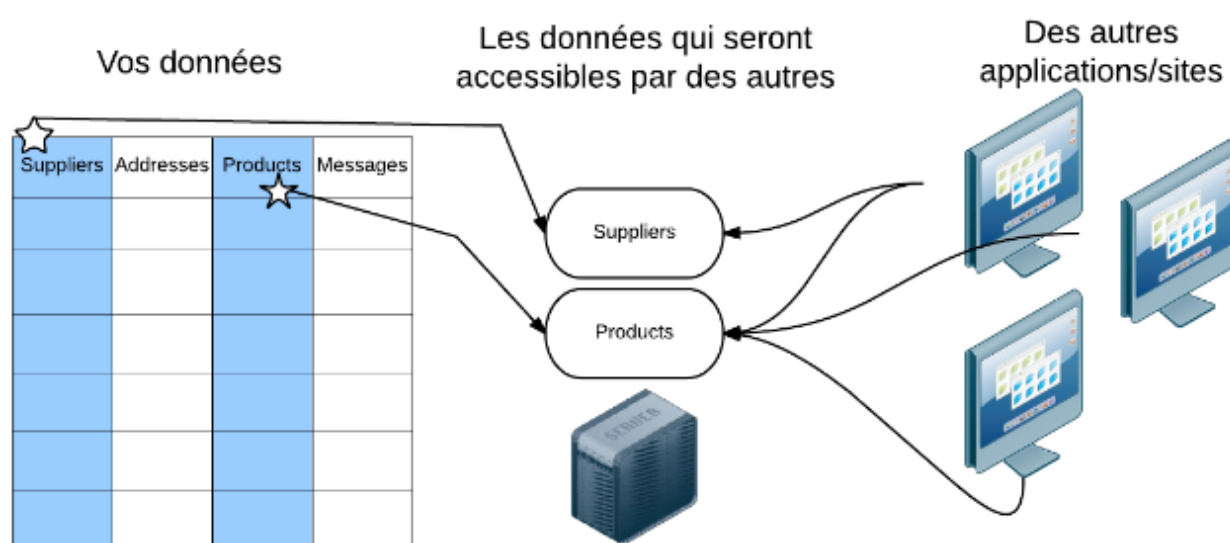


FIGURE 2.1. Illustration du fonctionnement d'une API

### 2.1.4 Les avantages à utiliser des APIs

#### 2.1.4.1 Pour le producteur

- Augmenter la portée de ses services en permettant à d'autres administrations ou entreprises d'interagir avec eux.
- Créer des marchés en facilitant l'accès aux données gouvernementales.
- Innover à coût minime en permettant à des acteurs externes de créer des usages en exploitant des API ouvertes pour des besoins non imaginés au départ.
- Répondre à coût marginal aux exigences réglementaires de transparence et d'échange de données de plus en plus fortes (loi DCRA, Dites-le Nous Une Fois...).

#### 2.1.4.2 Pour le consommateur

- Proposer des services plus complets à l'utilisateur tout en diminuant la saisie d'informations.
- Économiser en mutualisant des briques logicielles.

### 2.1.5 Exemples d'API

#### 2.1.5.1 Windows API

**Windows API** est une collection de fonctions, de types de données et de constantes, en langage de programmation C (alors que Windows a été essentiellement écrit en Pascal...), qui permet à des programmeurs de créer des applications pour les systèmes d'exploitation Windows. Elle offre la possibilité de manipuler des fichiers, des processus, communiquer par les réseaux et manipuler des interfaces graphiques

#### 2.1.5.2 YouTube API

**YouTube** est un site web d'hébergement de vidéos et un média social sur lequel les utilisateurs peuvent envoyer, regarder, commenter, évaluer et partager des vidéos en streaming. Il a été créé en février 2005 par Steve Chen, Chad Hurley et Jawed Karim, trois anciens employés de PayPal, et racheté par Google en octobre 2006 pour 1,65 milliard de dollars. Le service est situé à San Bruno, en Californie.

L'api de YouTube permet aux sites web de pouvoir offrir à leurs visiteurs des vidéos en streaming depuis YouTube sans quitter leurs sites permettant ainsi de garder les internautes sur le site.

#### 2.1.5.3 Google Maps API

L' **API Maps Static** vous permet d'incorporer une image Google Maps sur votre page Web sans nécessiter JavaScript ni aucun chargement de page dynamique. Le service Maps Static API crée votre carte en fonction des paramètres d'URL envoyés via une demande HTTP standard et renvoie la carte sous forme d'image que vous pouvez afficher sur votre page Web.

## 2.2 Présentation des solutions existantes

### 2.2.1 Contexte

Nous amorçons actuellement la deuxième décennie du vingt et unième siècle, sachant que la grande révolution de l'informatique date du début des années soixante-dix, nous nous trouvons pratiquement à un demi-siècle de celle-ci, cette période aura contribué à l'avènement de nouvelles et nombreuses technologies et à la création de multiples applications, même si l'implémentation des API est une solution qui existe depuis un temps relativement court, dans sa version actuelle cela constitue tout de même un temps considérable en informatique où les technologies évoluent rapidement. C'est donc ainsi que des développeurs ont déjà pu mettre sur pied bon nombre de solutions de ce type.

### 2.2.2 Solutions de relance

Une relance correspond à l'envoi d'une information à une ou plusieurs personnes par le biais de messages envoyés au travers de moyens de communication ou d'outils quelconques. Le besoin de communication étant un besoin essentiel à l'homme, et les systèmes informatiques étant basés principalement dessus, c'est donc normalement qu'ils offrent aujourd'hui un nombre incalculable de moyen d'effectuer des relances et que des services préconçus et assez complet offrent la possibilité d'ajouter assez facilement des moyens de relance aux développeurs de projets informatiques pour leurs permettre de gagner en rapidité dans la réalisations de leurs applications offrant en plus cette fonctionnalité. On peut citer parmi les solutions disponibles celles décrites à la suite :

#### 2.2.2.1 Gmail API

L'API de **Gmail** offre la possibilité aux développeurs d'intégrer à leurs applications un ensemble de fonctionnalités liées à Gmail leurs permettant ainsi d'offrir aux utilisateurs des fonctions tels que : la lecture et l'envoi de messages, la gestion des brouillons et des pièces jointes, la recherche des fils et des messages, le travail avec des libellés, la configuration des messages push et la gestion des paramètres Gmail.

#### 2.2.2.2 API Twilio pour WhatsApp

L'API Twilio offre la possibilité à ses utilisateurs d'envoyer des messages par plusieurs moyens parmi lesquels l'application WhatsApp, WhatsApp est une application de messagerie instantanée utilisée par environ 1,5 milliard de personnes à travers le monde. L'API de twilio

offrent ainsi d'envoyer des messages en temps réel, de recevoir des accusés de réception et les confirmations de lecture. Les messages WhatsApp sont chiffrés de Twilio vers l'appareil et sécurisés via HTTPS de votre application Twilio, permettant des conversations privées avec les utilisateurs.

### **2.2.2.3 API Nexmo pour SMS**

L'API Nexmo offre quant à elle la possibilité aux développeurs de pouvoir envoyer des messages et ceci de manière programmable afin de pouvoir avertir leurs clients au moment propice lorsque qu'un événement déterminant survient ou le moment venu.

Nous constaterons que les solutions de relances sont nombreuses et implémentées chacune pour des moyens divers et avec un fonctionnement qui lui aussi peut différer d'une API à l'autre mais nous précisons ici que la plupart des API présents sur le marché sont généralement payantes et le plus souvent adaptées aux grilles tarifaires de leurs régions d'origine.

### **2.2.3 Solutions d'échéancier**

Autant les solutions de relance pleuvent et semblent couler de source, autant pour ce qui est des solutions d'échéancier de facture, force est de constater que en ce qui concerne ces cas les solutions se font plus timides, là nous n'avons pas grand-chose à nous mettre sous la dent. Ceci s'explique certainement par le fait que les développeurs préfèrent sans doute chacun définir leur propre solution. Ce qui a pour conséquence la difficulté à trouver une solution idéale à la réalisation de cette fonctionnalité parmi les APIs disponibles, et pour ce qui est des solutions d'échéancier, il s'agit-là d'une fonctionnalité très souvent visible dans la vie de tous les jours mais très peu vulgariser par les acteurs du monde informatique.

## **2.3 Problématique**

Après avoir constaté l'absence d'une application présentant la solution à notre problème il devient assez clair que c'est nous qui allons devoir la mettre sur pied.

Quels Seront donc les moyens, les outils et les méthodes que nous devrons utiliser pour la résolution de notre problème ?

Après notre étude des solutions existantes, des différentes articulations de notre problème nous pouvons retenir que des débuts de solutions existent déjà mais présentent les limites suivantes :

- les solutions existantes ne sont pas complètes dans la mesure où il n'existe aucune solution permettant seule de résoudre notre problème dans son entièreté ;
- les solutions existantes ont un coût qui n'est pas toujours en adéquation avec nos grilles tarifaires ; et
- il existe des fonctionnalités pour lesquelles nous ne sommes pas en mesure de trouver des solutions déjà implémentées.

## **2.4 Solutions apportées**

Après avoir constaté l'absence de solution adéquate, il sera question pour nous de chercher à concevoir une solution répondant à nos besoins. Nous ne partirons pas de zéro tout de même vu que certaines fonctionnalités souhaitées sont déjà apportées par des API disponibles.

Pour toutes ces raisons il est évident que pour la conception et la réalisation de notre solution nous devrions nous appuyer sur les solutions déjà existantes dès que celles-ci correspondraient à nos attentes et à y ajouter des fonctionnalités nouvelles et même inédites. C'est ainsi que nous pouvons nous pencher dès lors sur l'analyse et la conception de notre système.

## **2.5 Conclusion**

Dans ce chapitre il a été question pour nous de faire une présentation des solutions existantes, de leurs limites et par la suite d'annoncer les fondements des solutions que nous comptons apporter pour remédier à celles-ci. C'est ainsi que dans le prochain chapitre nous allons nous attaquer à l'analyse du problème et à la conception de la solution.

## ANALYSE ET CONCEPTION

Dans ce chapitre nous allons nous concentrer sur l'analyse et la conception de l'ensemble de notre système en nous appuyant sur ses acteurs et les fonctionnalités voulues.

### Contents

---

3.1	Analyse . . . . .	<b>15</b>
3.1.1	Définition des termes clés . . . . .	15
3.1.2	Cahier de charge . . . . .	15
3.1.3	Les acteurs . . . . .	16
3.1.4	Les cas d'utilisations . . . . .	17
3.1.5	Description des cas d'utilisations . . . . .	17
3.2	Description textuelle des cas d'utilisations principaux . . . . .	<b>18</b>
3.2.1	Définir un échéancier . . . . .	18
3.2.2	Enregistrer un paiement . . . . .	18
3.2.3	Enregistrer une relance . . . . .	19
3.3	Cas d'utilisations internes au système . . . . .	<b>19</b>
3.3.1	Contrôler les échéanciers . . . . .	19
3.3.2	Faire une relance . . . . .	20
3.4	Conception . . . . .	<b>21</b>
3.4.1	Diagramme de cas d'utilisations . . . . .	21



---

3.4.2	Diagramme de séquence . . . . .	22
3.4.3	Diagramme de classe . . . . .	23
3.5	Conclusion . . . . .	<b>23</b>

---

## 3.1 Analyse

### 3.1.1 Définition des termes clés

Dans le contexte de notre projet les mots ci-dessous doivent avoir les connotations suivantes :

- **Produit** : bien fini ou semi-fini par élaboré l'activité humaine à partir de matière première ;
- **Gestion** : action de s'occuper du bon déroulement des opérations ;
- **Échéanciers** : un registre de paiement à effectuer à la date de leur échéance ;
- **Échéance** : date à laquelle expire un délai ;
- **Relance** : un rappel fait à une personne à but informatif ;
- **Partenaire** : personne qui achète ou requière des services moyennant rétribution.

### 3.1.2 Cahier de charge

#### 3.1.2.1 Déroulement nominal

Un produit ou plusieurs produits sont réservés au nom d'un client dans une réservation, le client a ensuite le choix pour la méthode de paiement : il peut soit payer directement l'intégralité de sa facture en une fois ou celui-ci peut convenir de payer la dite facture en plusieurs fois, un échéancier de paiement est alors défini a son nom et comportant l'identifiant de la facture comptant une ou plusieurs échéances, comportant elles même chacune une ou plusieurs relances, une relance correspond ici à une note d'information envoyée par le biais d'un canal (WhatsApp, email, SMS, ou appel téléphonique) à un client pour lui indiquer à quel niveau il se trouve et des échéances de paiement futures. Une relance est faite tout d'abord aux dates prévues, mais aussi dans le cadre du manquement du paiement d'une relance ou de la modification des paramètres d'un échéancier créé par ces manquements.

#### 3.1.2.2 Fonctionnalités attendues

Il nous est donc demandé dans le cadre de notre projet de réaliser une API avec les fonctionnalités suivantes :

- **Gestion des échéances** : permet de définir un échéancier de paiement pour le paiement d'une facture et s'assurer que celui-ci est respecté ;
- **relance des partenaires** : permet l'envoi de messages aux clients pour les avertir de l'état d'avancement des paiements et des prochaines échéances.

En résumé le système devra donc :

- **permettre de gérer un échéancier :**
  - enregistrer un échéancier
  - enregistrer un paiement
- **permettre de relancer les partenaires :**
  - enregistrer une relance.

### 3.1.3 Les acteurs

Etant dans le cadre de la réalisation d'une API les acteurs qui devront souvent être amenés à utiliser notre application seront essentiellement d'autres applications.

- **Les acteurs principaux :** les applications consommatrices de notre API ;
- **Les acteurs secondaires :** les API auxquelles notre API aura recours.

La figure 3.1 ci-dessous présente le système comme une boîte noire avec autour tous les éléments en communication avec lui.

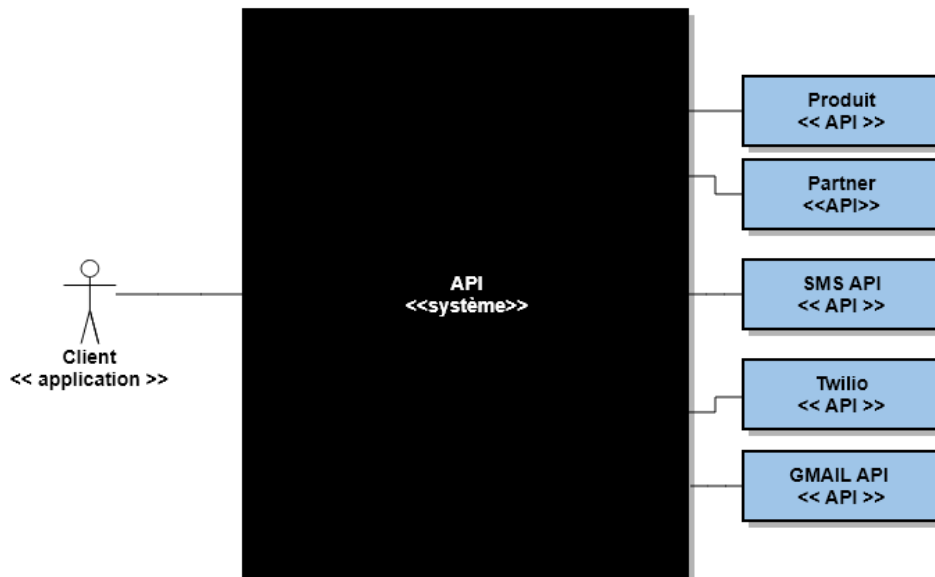


FIGURE 3.1. Diagramme de contexte

Les différents acteurs de type api agissent avec le système en lui offrant un ensemble de fonctionnalités tandis que le client exploite les fonctionnalités offertes par le système. L'exploitation de ces fonctionnalités est définie par les cas d'utilisations suivants.

### 3.1.4 Les cas d'utilisations

Les applications pourront solliciter notre système pour plusieurs raisons déjà évoquées ici, à savoir :

- **Gérer les échéanciers :**

1. définir un échéancier ()
2. enregistrer un paiement ()

- **Gérer les relances :**

1. enregistrer une relance ()

### 3.1.5 Description des cas d'utilisations

#### 3.1.5.1 Gérer les échéanciers

Le système permettra d'effectuer des opérations de création, lecture et de modification à la fois sur les échéanciers et les échéances mais n'a pas la possibilité de les supprimer, à la place le système peut leur passer le statut annulé. Le système permettra en plus et surtout de :

- **définir un échéancier ()**
- **enregistrer un paiement ()**

#### 3.1.5.2 Gérer les relances

Le système permettra de réaliser les opérations de lecture, modification et suppression sur les relances il permettra en plus et surtout d'enregistrer des modèles de relance et devra envoyer des messages aux clients pour les tenir au courant de l'état de leurs échéanciers et des prochaines dates de règlement, ainsi que les changements dans les paramètres de leurs échéanciers(ajout de pénalités). Mais on retiendra comme fonctionnalité principale offerte à ce niveau :

- **enregistrement d'une relance ()**

## 3.2 Description textuelle des cas d'utilisations principaux

### 3.2.1 Définir un échéancier

- **Nom du cas d'utilisation** : définir échéancier ;
- **Acteur principal** : application cliente ;
- **Description** : permet au système d'enregistrer un échéancier ;
- **Enchaînement nominal** :
  1. L'application fournit les informations nécessaires (date, montant, échéancier, etc. . . ) ;
  2. Le système numérote l'échéance ;
  3. Le système enregistre l'échéance ;
    - si il y'a déjà une échéance en cours : on met la nouvelle échéance en attente ;
    - sinon on met l'échéance en cours ;
- **Déroulement alternatif** :
  - Si l'application ne fournit pas d'échéancier :
    1. le système crée un échéancier ;
    2. le système numérote échéance comme première échéance ;
    3. le système enregistre l'échéance à l'échéancier
    4. l'état de l'échéance est alors « en cours » de règlement.

### 3.2.2 Enregistrer un paiement

- **Nom du cas d'utilisation** : enregistrer paiement ;
- **Acteur principal** : application cliente ;
- **Description** : permet au système de renseigner le paiement d'un échéancier ;
- **Pré-condition** : enregistrer échéancier ;
- **Enchaînement nominal** :
  1. L'application fournit le montant et l'échéancier ;
  2. Le système va chercher dans l'échéancier l'échéance en cours ;
  3. Il le règle du montant renseigné ;
  4. Le système contrôle l'échéancier ;
- **Post-condition(s)** :

- Le règlement est effectué ;
- Le l'échéancier est contrôlé.

### 3.2.3 Enregistrer une relance

- **Nom du cas d'utilisation** : enregistrer relance ;
- **Acteur principal** : application cliente ;
- **Description** :
  1. permet d'ajouter un modèle relance ;
  2. permet d'ajouter une relance complète.
- **Enchaînement nominal** :
  1. L'application fournit : un moyen, un message, l'échéancier, le client et la date ;
  2. Le système enregistre la relance ;
  3. Le système fait la relance ;
- **Enchaînement alternatif** :
  - Si l'application ne fournit qu'un moyen de transmission et un message : le système enregistre un modèle de relance
- **Post-condition(s)** :
  - Le modèle de relance est enregistré ;
  - La relance est enregistrée.

## 3.3 Cas d'utilisations internes au système

### 3.3.1 Contrôler les échéanciers

- **Nom du cas d'utilisation** : contrôler échéanciers ;
- **Acteur principal** : système interne ;
- **Description** : permet de contrôler les états des échéanciers ;
- **Pré-condition** : enregistrer échéancier ;
- **Enchaînement nominal** :
  1. Le système parcourt toutes les échéances de tous les échéanciers ;
  2. Le système récupère toutes les échéances en cours ;
  3. Le système vérifie les dates de règlement ;
  4. Si la date de règlement de l'échéance est dépassée :

- a) Le système change le statut de l'échéance en pénalisé;
  - b) S'il y'a une prochaine échéance :
    - i. le système passe à l'échéance suivante;
    - ii. le système lui ajoute la somme de l'échéance précédente puis avec les Pénalités convenues pour les cas de non-paiement dans les délais
    - iii. le système fait une relance.
  - 5. Si il n'y a pas de prochaine échéance :
    - a) Le système crée une nouvelle échéance;
    - b) Le système passe à l'échéance suivant;
    - c) Le système lui ajoute la somme de l'échéance précédente puis avec les pénalités convenues pour les cas de non-paiement dans les délais;
    - d) le système fait une relance.
- **Post-condition(s)** :
- Le contrôle est effectué;

### 3.3.2 Faire une relance

- **Nom du cas d'utilisation** : faire relance;
- **Acteur principal** : système interne;
- **Description** : permet d'envoyer une relance à un client;
- **Pré-condition** : enregistrer relance;
- **Enchaînement nominal** :
  - 1. L'application fournit un modèle de relance, un échéancier, le client, et la date;
  - 2. Le système enregistre la relance;
  - 3. Le moment venu le système envoie la relance;
- **Déroulement alternatif** :
  - Le système peut lui-même envoyer des messages en cas de survenue d'un événement pertinent;
  - Le système renseigne alors uniquement le modèle de relance, le client, et l'échéancier, le message est alors envoyé immédiatement.
- **Post-condition(s)** :
  - La relance est effectuée;

## 3.4 Conception

### 3.4.1 Diagramme de cas d'utilisations

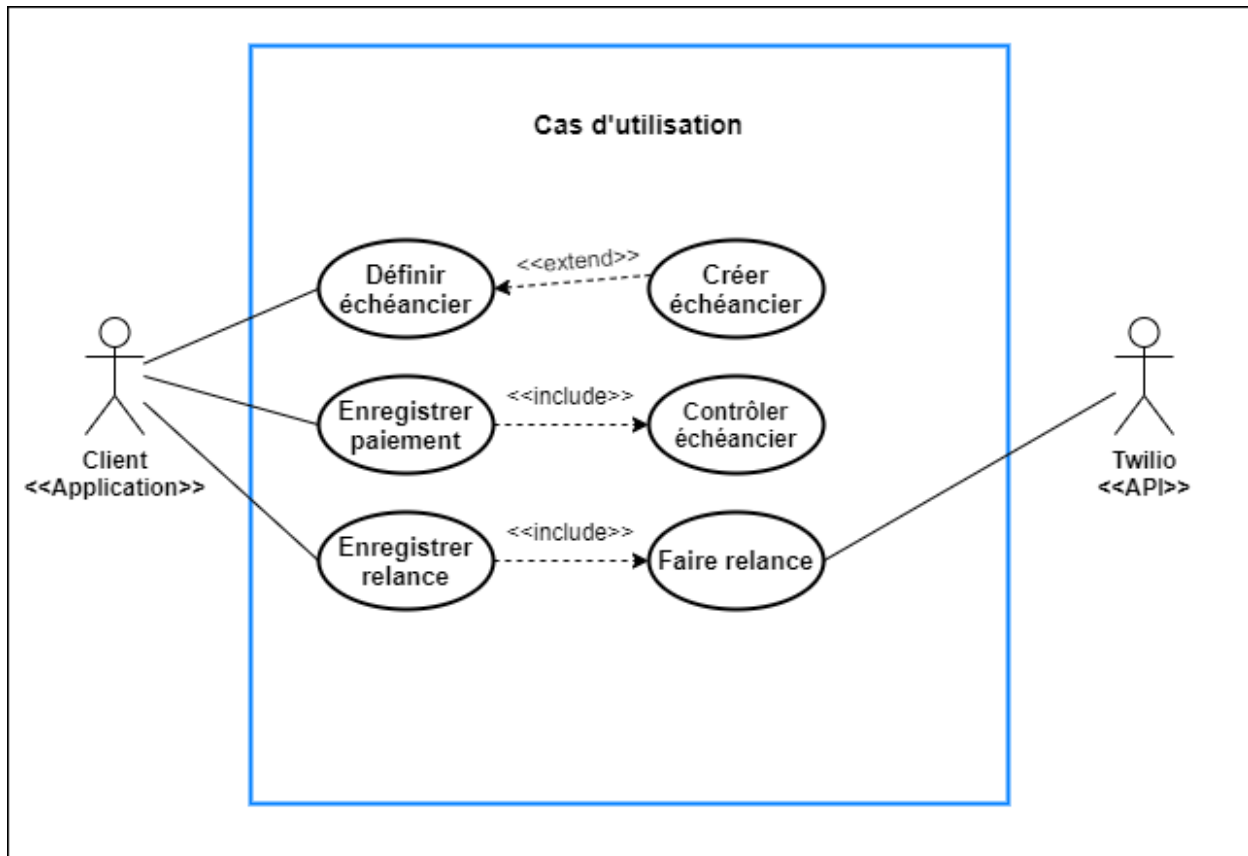


FIGURE 3.2. Diagramme de cas d'utilisation



### 3.4.2 Diagramme de séquence

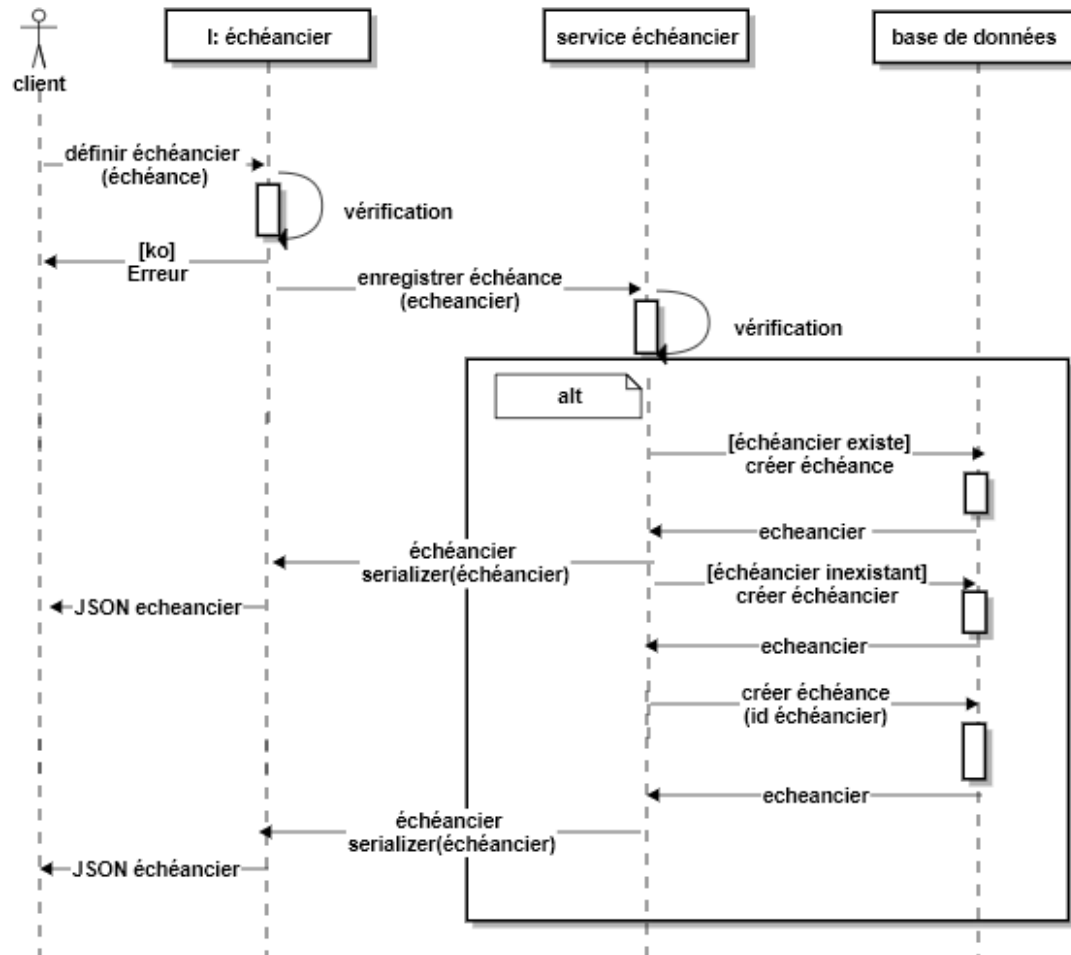


FIGURE 3.3. diagramme de séquence (enregistrer échéancier)

### 3.4.3 Diagramme de classe

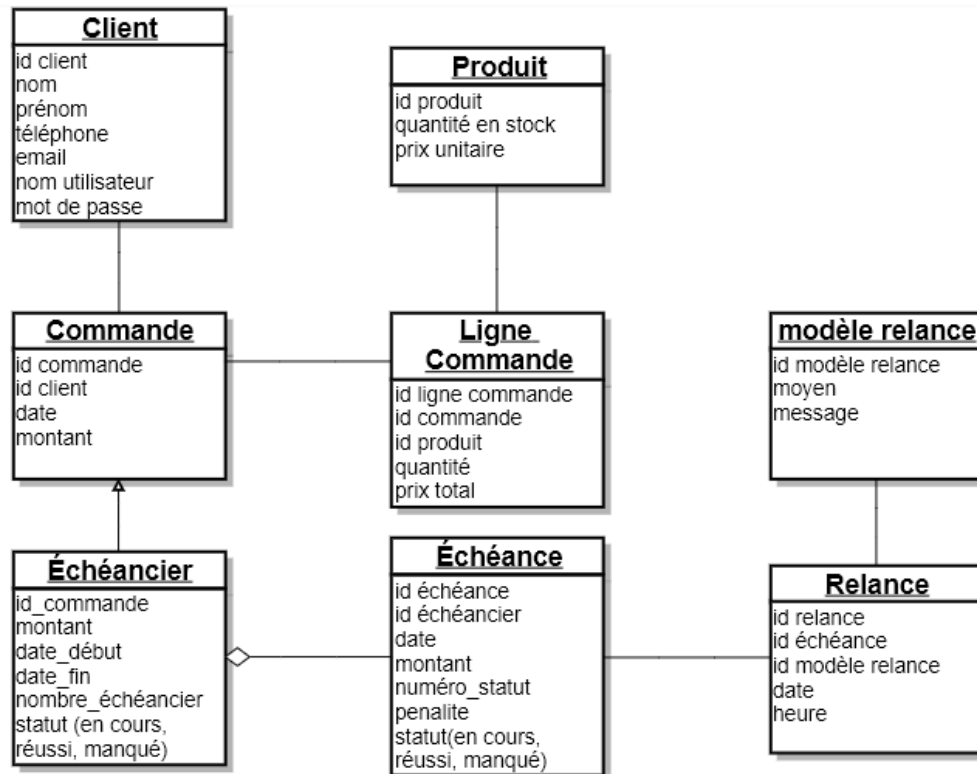


FIGURE 3.4. diagramme de classe

## 3.5 Conclusion

Parvenus au terme à la fois de notre analyse et de notre conception, nous avons à présent assez de matière pour passer à l'implémentation, après avoir au préalable réuni tous les outils nécessaires.

## IMPLEMENTATION ET RESULTAT FINAL

Dans ce chapitre il sera question pour nous de présenter les grands outils utilisés et le résultat final obtenu au terme de la réalisation.

### Contents

4.1	Les outils utilisés pour la création de notre système . . . . .	<b>25</b>
4.1.1	Python . . . . .	25
4.1.2	Django . . . . .	25
4.1.3	Django Rest Framework . . . . .	25
4.1.4	ReactJS . . . . .	26
4.2	Quelques vues de la réalisation . . . . .	<b>27</b>
4.2.1	Page d'authentification . . . . .	27
4.2.2	Page d'accueil . . . . .	27
4.2.3	Page Nouveau Contrat . . . . .	28
4.2.4	Page Client . . . . .	30
4.3	conclusion . . . . .	<b>30</b>

## 4.1 Les outils utilisés pour la création de notre système

### 4.1.1 Python

Python est un langage de programmation interprété, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions ; il est ainsi similaire à Perl, Ruby, Scheme, Smalltalk et Tcl. Le langage Python est placé sous une licence libre proche de la licence BSD7 et fonctionne sur la plupart des plates-formes informatiques, des smartphones aux ordinateurs centraux<sup>8</sup>, de Windows à Unix avec notamment GNU/Linux en passant par macOS, ou encore Android, iOS, et peut aussi être traduit en Java ou .NET. Il est conçu pour optimiser la productivité des programmeurs en offrant des outils de haut niveau et une syntaxe simple à utiliser. Il est également apprécié par certains pédagogues qui y trouvent un langage où la syntaxe, clairement séparée des mécanismes de bas niveau, permet une initiation aisée aux concepts de base de la programmation.

### 4.1.2 Django

Django est un cadre de développement web open source en Python. Il a pour but de rendre le développement web 2.0 simple et rapide. Pour cette raison, le projet a pour slogan « Le Framework pour les perfectionnistes avec des deadlines. ». Développé en 2003 pour le journal local de Lawrence (Kansas), Django a été publié sous licence BSD à partir de juillet 2005 [7].

### 4.1.3 Django Rest Framework

Le Framework Django REST est une boîte à outils puissante et flexible pour la création d'API Web.

Quelques bons points du Framework REST :

- L'API navigable sur le Web est un énorme gain de convivialité pour les développeurs ;
- Stratégies d'authentification, y compris les packages pour OAuth1a et OAuth2 ;
- Sérialisation prenant en charge les sources de données ORM et non ORM ;
- Personnalisable tout le chemin vers le bas - il suffit d'utiliser des vues sur la base de la fonction réguliers si vous n'avez pas besoin des plus puissantes fonctionnalités ;
- Documentation complète et grande assistance communautaire ; et
- Utilisé et approuvé par des sociétés internationalement reconnues telles que Mozilla , Red Hat , Heroku et Eventbrite.

### 4.1.4 ReactJS

ReactJS est une bibliothèque JavaScript libre développée par Facebook depuis 2013. Le but principal de cette bibliothèque est de faciliter la création d'application web monopage, via la création de composants dépendant d'un état et générant une page (ou portion) HTML à chaque changement d'état. React est une bibliothèque qui ne gère que l'interface de l'application, considéré comme la vue dans le modèle MVC. Elle peut ainsi être utilisée avec une autre bibliothèque ou un framework MVC comme AngularJS. La bibliothèque se démarque de ses concurrents par sa flexibilité et ses performances, en travaillant avec un DOM virtuel et en ne mettant à jour le rendu dans le navigateur qu'en cas de nécessité. La bibliothèque est utilisée par Netflix<sup>3</sup> (côté serveur uniquement depuis le 25 octobre 2017 pour gagner 50 % de performance<sup>4</sup>), Yahoo, Airbnb, Sony, Atlassian ainsi que par les équipes de Facebook, appliquant le dogfooding sur le réseau social éponyme, Instagram ou encore WhatsApp. À la fin de 2015, WordPress.com annonce Gutenberg, une interface pour les éditeurs de sites WordPress, développée en JavaScript avec Node.js et React.

## 4.2 Quelques vues de la réalisation

KSM Dateliners est une application développée en Python côté serveur et ReactJS côté client, qui permet la gestion des échéanciers et des relances des clients de diverses sociétés.

### 4.2.1 Page d'authentification

Pour accéder à l'application, l'utilisateur a besoin de s'authentifier. Pour cela il doit remplir les champs nom d'utilisateur et mot de passe.

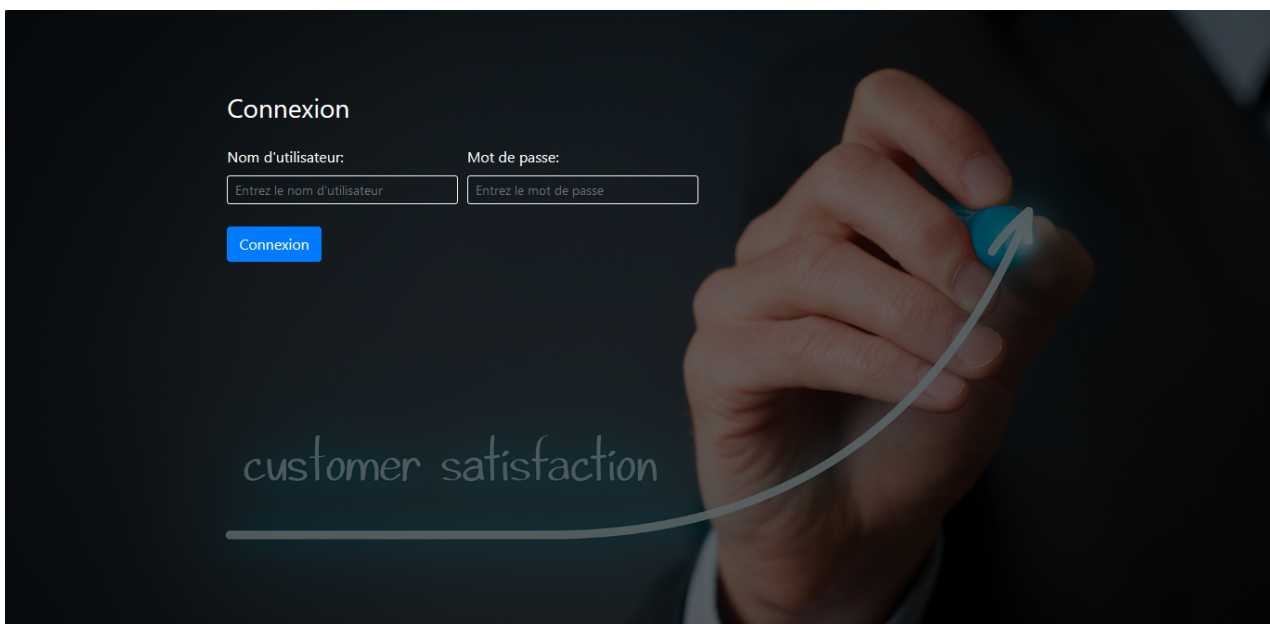


FIGURE 4.1. Page d'authentification

### 4.2.2 Page d'accueil

Après s'être authentifié avec succès, l'utilisateur accède enfin à la page d'accueil de l'application. Dans laquelle, il lui est présenté les différentes fonctionnalités de l'application à savoir :

1. La création des contrats de créances ;
2. La gestion des échéances ;
3. La gestion des relances.

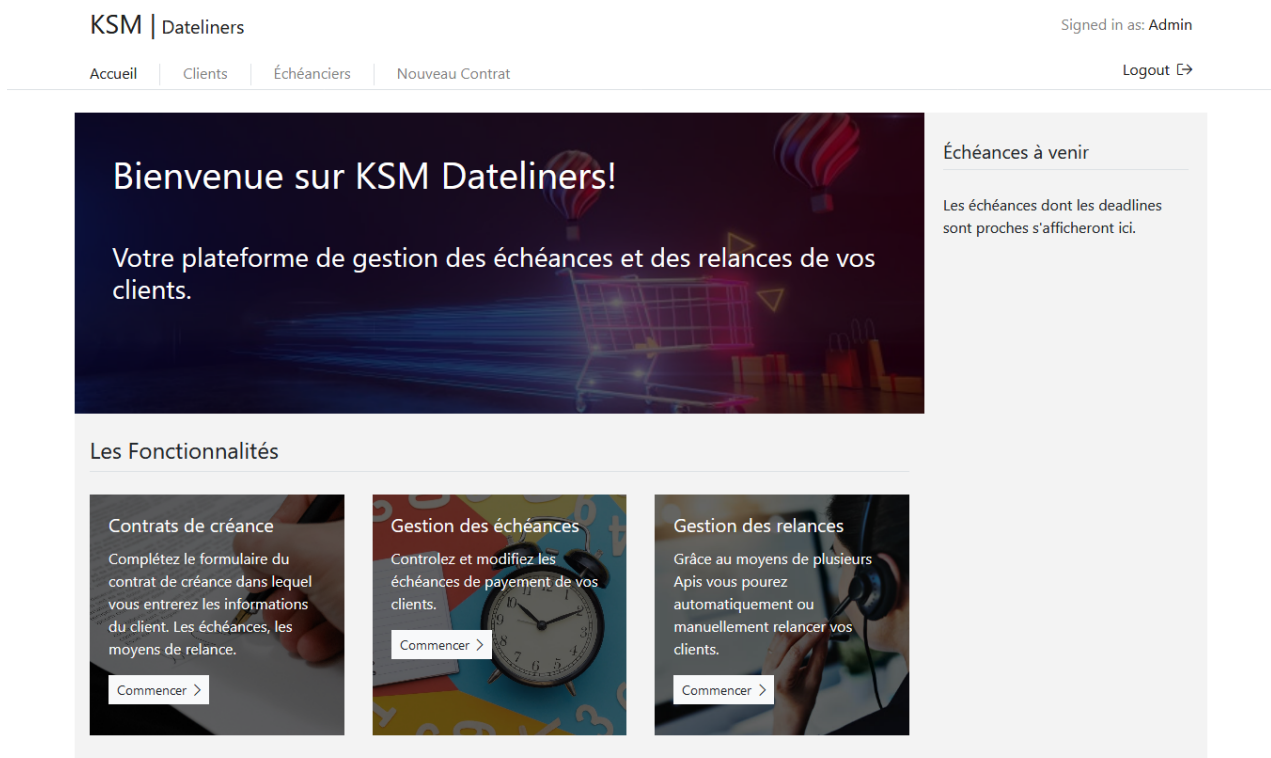


FIGURE 4.2. Page d'accueil

### 4.2.3 Page Nouveau Contrat

C'est dans cette page que l'administrateur saisit les informations nécessaires pour l'accord de créance entre la société et le client

Il lui est d'abord présenté les différentes consignes de remplissage :

- Remplir tous les champs de haut en bas de la gauche vers la droite ;
- Choisir la devise (monnaie) ou laisser XAF qui correspond au Franc CFA de la CEMAC ;
- La devise choisie sera la même pour tous les montants (échéances, garants) ;
- Entrer le nombre d'échéances, une liste de champs va être déroulée correspondants aux détails de chacune des échéances qu'il faudra remplir ;
- La somme des montants des échéances doit être exactement égale au montant de la dette ;
- Choisir les moyens de relance ou laisser les moyens par défaut ;

- Entrer au moins un élément garant de telle sorte que sa valeur estimée soit au moins égale à la valeur de la dette ;
- Imprimer le contrat, le faire signer par les parties ; et
- Soumettre le formulaire.

### Accord entre Parties

[Consignes de remplissage](#) >

#### Informations Générales


Client / Créancier\* :

Motif\* :

Montant dette\* :

Montant en Lettres :

Payable en\* :

Date Butoire\* :    
Cliquez pour sélectionner la date.

Taux de peinalité\* :

#### Moyens et Priorités des Relances

Relance 1\* :

Relance 2\* :

Relance 3\* :

Relance 4\* :

#### Garants de la dette

[Ajouter un élément garant +](#)

[Annuler](#) [Envoyer](#)

FIGURE 4.3. Page Nouveau Contrat



#### 4.2.4 Page Client

Pour gérer les échéanciers, contrôler les échéances et relancer les clients. L'administrateur doit tout d'abord sélectionner un client

- Sélectionner un client ;
- Filtrer la liste des échéanciers en fonction des besoins ;
- Sélectionner l'échéancier
- Imprimer la liste des échéances/ afficher les détails ;

KSM | Dateliners Signed in as: Admin

Accueil | Clients | Échéanciers | Nouveau Contrat Logout ↗

---

Payement / Règlement des échéances

Client / Créancier\* :

Charger les créances : ☒ Toutes ☐ En cours ☐ Soldées  
☐ Avec pénalité

Résultats: (0) échéancier(s) trouvé(s)

FIGURE 4.4. Page Client

### 4.3 conclusion

En somme, ce chapitre nous a permis de présenter les outils qui nous ont servis à implémenter notre solution et le résultat obtenu après la réalisation de la solution à notre problème.

## CONCLUSION

En définitive, Il était question pour nous dans le cadre de notre travail de mettre sur pied une solution permettant à la fois de gérer un échéancier de paiement ainsi que des relances à des partenaires, ceci sous la forme d'une API. Pour y arriver, nous avons mis en pratique nos connaissances apprises à l'école :

Tout d'abord nous avons mené une étude de l'existant dans le monde qui nous permettrait de résoudre notre problème de manière partielle ou globale.

En fonction de l'étude précédente, nous avons choisi la meilleure approche dans la résolution de notre problème.

En fonction de la solution choisie, nous avons mené une analyse du problème en faisant ressortir les contraintes techniques du problème.

Ensuite, nous avons modélisé notre problème à l'aide de l'UML, au moyen du diagramme des cas d'utilisations (pour nous permettre de représenter les services offerts par notre solution), des classes (pour représenter les différentes classes).

Pour finir, nous avons à partir de la modélisation, implémenté notre solution à l'aide des langages Python et ReactJS, des frameworks Django, Django REST.

Les différentes phases ci-dessus nous ont permis de remplir le cahier de charges en respectant les exigences qui y sont inscrites et de fournir un travail à la hauteur de la qualité des enseignements au sein de l'école polytechnique.

Enfin, le stage au sein du CETIC a été très bénéfique pour moi en ce sens qu'il m'a permis de découvrir le style d'architecture REST et ses performances sur le Web, de me familiariser avec la technologie ReactJS et l'API Twilio.

## **BIBLIOGRAPHIE**

1. [https://fr.wikipedia.org/wiki/Interface\\_de\\_programmation](https://fr.wikipedia.org/wiki/Interface_de_programmation)  
⇒ Interface de programmation
2. [https://fr.wikipedia.org/wiki/Application\\_\(informatique\)](https://fr.wikipedia.org/wiki/Application_(informatique))  
⇒ Application (informatique)
3. [https://fr.wikipedia.org/wiki/Django\\_\(framework\)](https://fr.wikipedia.org/wiki/Django_(framework))  
⇒ Django (framework)
4. <https://www.django-rest-framework.org/>  
⇒ Django REST (framework)
5. <https://www.supinfo.com/articles/single/5642-qu-est-ce-quune-api-rest-restful>  
⇒ API RESTFUL