

Documentation Stage Salesforce

Ryan El Fatihi

July 2024

Contents

1	Vision Globale du Projet	2
2	Création des Tables	3
3	Création des Prompts sans RAG	4
4	Flows	6
5	Explication et Configuration RAG	7
6	Prompts avec RAG	7
7	Configuration de l'API Google	7
8	Explication du Script Python	9
9	Création du Composant RFP	12
10	Annexe	13

1 Vision Globale du Projet

Chez Salesforce, la réception fréquente d'appels d'offres (RFP) nécessite des réponses souvent fastidieuses, répétitives et à faible valeurs ajoutées bien qu'essentiels pour initier des deals. Une automatisation de ce document pourrait alors être très utile et permettrait d'améliorer l'efficacité et la valeur ajoutée du travail des Solution Engineer (SE). L'objectif de ce document est donc de réaliser une esquisse de ce à quoi pourrait ressembler une automatisation de ce processus.

Cette automatisation va se réaliser en plusieurs étapes, tout d'abord il faudra récupérer les informations liées au RFP étudié. Il faudra aussi générer les différentes informations pertinentes à une réponse. Enfin il faudra récupérer ces réponses et les générer dans des googles slides, plus précisément dans des zones de texte qui vont bien.

On suivra le schéma suivant :



Figure 1: Schema global du projet

0. Utilisation de la technologie RAG pour vectoriser de grandes quantités de données.
1. Développement de composants pour recueillir les différentes informations liées au RFP.
2. Génération des différentes informations importantes pour répondre aux RFP.
3. Gestion de la génération des Google Slides à l'aide des APIs Google.

Les bénéfices attendus de cette automatisation incluent une meilleure répartition du temps des Solutions Engineers (SE), augmentant ainsi la valeur ajoutée de leur travail. De plus, l'utilisation des outils Salesforce permettra une intégration facile dans le CRM existant. Bien que l'automatisation ne pourra pas encore générer totalement la réponse au RFP, elle réduira considérablement le temps nécessaire pour préparer une réponse initiale.

2 Création des Tables

Il est très important de stocker toutes les données que l'on va utiliser ainsi que celle qui vont être générées. C'est pourquoi l'utilisation de tables est importante. Je vais préciser les différentes tables que j'ai créées dans un premier temps, cette liste est non exhaustive et peut varier. Elle a pour but de permettre une première reconstitution fonctionnelle dans Salesforce et de faciliter la compréhension de l'enjeu de ces différentes tables.

Objet	Champs
RFP	Contenu
Executive Summary	Enjeux, Strategie, Why Salesforce
Solutions Salesforce	Solutions prompt, Solutions Json
Success Story	Temoignage

Table 1: Objets Salesforce et champs associés

Explication de chacun des champs :

Contenu : Contient le contenu texte du RFP

Enjeux : Contient les différents enjeux liés à l'appel d'offre

Stratégie : Contient la stratégie à mettre en place afin de répondre aux différents enjeux et problématiques du client.

Why Salesforce : Contient les raisons qui poussent le client à choisir Salesforce et pas un autre concurrent

Solutions prompt : Contient un texte exposant les différentes solutions et les raisons qui font qu'elles sont adaptées au client et à son appel d'offre

Solutions Json : Contient le résultat de Solutions prompt dans un format Json adapté et mieux structuré

Temoignage : Contient le témoignage d'une success story customer (trouvé sur le site help.salesforce.com)

3 Création des Prompts sans RAG

Afin de répondre avec texte cohérent avec l'appel d'offre, il faut utiliser de l'IA générative. Salesforce propose cela avec notamment l'utilisation de prompt builder. Ces prompts vont permettre de remplir les différents champs pour chaque record. On s'intéressera dans cette partie uniquement aux prompts n'utilisant pas la vectorisation fournie par le RAG.

Aussi, cette partie regroupera l'ensemble des prompts (en leurs états actuels, susceptible d'évoluer pour être plus précis et pour éviter les hallucinations). On notera enfin que pour chacun des prompts, le modèle choisi est le suivant : "Anthropic Claude 3 Haiku on Amazon". Le choix de ce modèle sera justifié en annexe.

Prompt "Secteur from RFP" :

La chose la plus importante : cite des exemples concrets d'utilisation pour !*Input : RFP.Name* Voici ci-dessous différentes solutions proposées par Salesforce. Peux-tu, en te basant très précisément sur les besoins du client présents dans !*Input : RFP.Contenu_c* (les citer), expliquer quelles sont les 3 solutions les plus adaptées au client et à ses problèmes. Aussi, la réponse se fera de la manière suivante : Nom de la solution suivie de l'explication se basant sur les besoins du client (précisément en les citant). Aussi donne les moi par ordre d'importance. Relie chacune des solutions que tu proposes à des objectifs fixés dans l'appel d'offre donné ci-dessus.

Sales Cloud, Service Cloud, Marketing Cloud, Commerce Cloud, Tableau, Mulesoft, Slack, Salesforce Einstein.

Aussi, je souhaiterais que tes réponses restent assez courtes (1 phrase par solution).

Prompt "RFP to Strategie" :

En utilisant le document suivant : !*Input : Exec_summary.Document1_c* ainsi que !*Input : Exec_summary.Enjeux_c*

Donne-moi précisément la stratégie qu'on adopterait pour répondre aux 3 enjeux majeurs de cet appel d'offre. Sois convaincant et pas trop long, et ne sois pas redondant. Utilise les technologies de Salesforce pour y répondre, mais je veux que les !*Input : Exec_summary.Enjeux_c* s'affichent tel quel.

Prompt "RFP to why Salesforce" :

En utilisant le document suivant : !*Input : Exec_summary.Document1_c*

Ainsi que les informations suivantes : !*Input : Exec_summary.Enjeux_c* et !*Input : Exec_summary.Strategie_c*

Décris de manière concise pourquoi Salesforce est LA BONNE entreprise

pour répondre à cet appel d'offre. Ne sois pas redondant.

Prompt "Solutions JSON from RFP" :

Peux tu à partir de ceci : *!Input : Solutions.Solutions_prompt_c* me générer un fichier JSON qui prendra chacune des solutions sous forme de clé et me renverra la description associée dans le fichier JSON. N'invente rien utilise seulement ce que je t'ai donné et ne génère pas de texte en dehors du fichier JSON. Ne génère pas de texte en dehors du fichier JSON. Le fichier Json devra être de la forme : "Clés":".", "Clé2":".", "Clé3":".". Les clés du fichier JSON ne pourront être que : Sales Cloud, Service Cloud, Marketing Cloud, Commerce Cloud, Tableau, Mulesoft, Slack, Salesforce Einstein Aussi je veux que le texte reste au plus proche possible de l'entreprise étudiée. Calcule le nom de l'entreprise : *!Input : RFP2.Name* à chaque fois dans le fichier JSON.

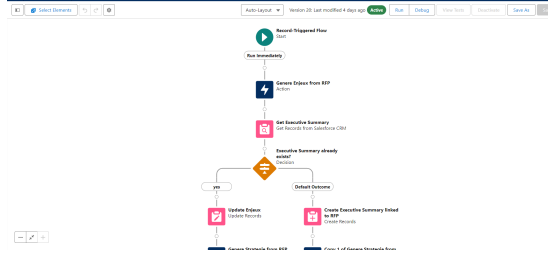
Prompt "Enjeux from RFP" :

Voici un appel d'offre : *!Input:RFP2.Contenu_c* Peux tu m'analyser ce document et me fournir à l'aide d'une phrase concise les Enjeux de l'entreprise faisant cet appel d'offre. Peux tu me générer cela et uniquement cela (pas de texte superflu en dehors du fichier JSON) sous la forme d'un fichier JSON de la forme : "Enjeu1" : ".","Enjeu2" : ".","Enjeu3" : "." Il faut qu'il n'y ait rien en dehors du fichier JSON. Fais le en peu de mots et sois concis et précis.

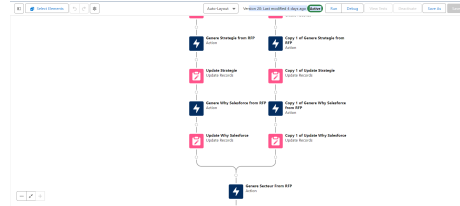
(On notera que ces prompts peuvent être sujet à des améliorations et qu'ils sont loin d'être parfaits)

4 Flows

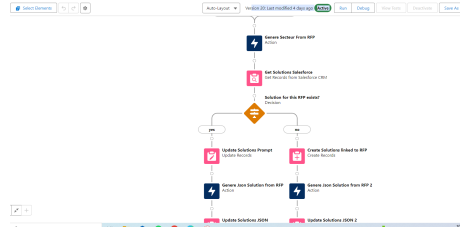
La génération des informations dans les différentes tables ainsi que leurs créations se fera de manière automatisées en récupérant les informations générées par les différents prompts à l'aide de flow.



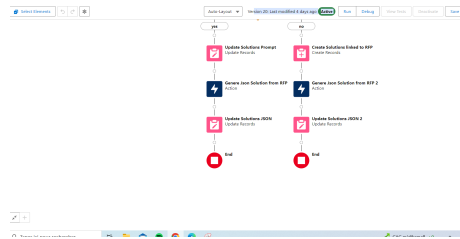
(a) Vision globale du flow 1



(b) Vision globale du flow 2



(c) Vision globale du flow 3



(d) Vision globale du flow 4

Figure 2: Vision globale des différents flows

5 Explication et Configuration RAG

De nombreuses données sont utiles pour générer de l'information pertinente et bien en lien à la fois avec l'appel d'offre étudié mais aussi avec Salesforce, ses différentes solutions ainsi que ses différents deals effectués dans le passé. Cependant la fenêtre d'input du prompt builder compte un nombre limité de token à utiliser.

C'est pourquoi on s'intéresse au RAG (Retrieval augmented Generation). Le RAG consiste en la vectorisation de grandes quantités de données afin qu'on puisse insérer uniquement la partie reliée à ce qui est demandée dans le prompt en sélectionnant le bon chunk. Je mets ci-dessous le lien d'une vidéo montrant plus précisément comment configurer et insérer une classe Apex directement dans le prompt:

"Lien Video RAG"

6 Prompts avec RAG

Des prompts sans le RAG sont très utiles dans la plupart des cas car la fenêtre contextuelle du prompt builder est assez grande. Cependant lorsqu'on souhaite utiliser une grande quantité d'informations, la technologie RAG devient indispensable. Voici deux use cases qui pourraient être fait dans le futur, pour générer des informations plus précises et plus adaptées au RFP étudié :

Trouver une bonne customer success Story parmi un grand ensemble. En effet le RAG peut permettre dans ce cas de trouver aisément la success story la plus adaptée à notre client faisant son RFP.

Utiliser le site [help.Salesforce.com](https://help.salesforce.com) pour chaque réponse technique lié au RFP.

Il est important de rappeler tout de même les limites de la technologie RAG, celle-ci ne peut pas servir à créer un résumé d'un grand ensemble de données.

7 Configuration de l'API Google

Une fois toutes les informations générées dans Salesforce, il nous reste un travail d'affichage à faire. En effet, on aimerait obtenir la réponse à l'appel d'offre sous forme de Google slides.

Pour cela et pour rendre le processus automatique, il faudra faire bon usage des APIs fournies par Google. Cependant l'utilisation de ces APIs nécessite la création d'un compte qui fournira un token permettant un très grand nombre d'utilisation.

Voici un petit guide expliquant les différentes étapes à effectuer pour obtenir ce token :

Rendez-vous sur ce lien :

[lien vers le google cloud console](#)

Une fois sur ce lien, connectez vous avec votre adresse mail (qui doit être une adresse gmail) et faites les démarches pour profiter d'un accès gratuit. Notez que cet accès est sans engagement et dure 3 mois, il ne se renouvelle pas automatiquement.

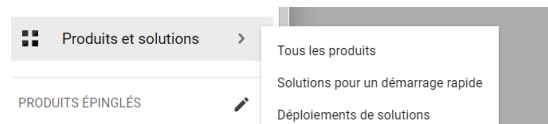
Une fois que cela est fait, créez un nouveau projet et allez dans l'onglet en haut à gauche, sélectionnez alors :

Produits et Solutions

Nos Produits

API & Services

Bibliothèques



(a)



(b) Configuration compte API google

Puis, cherchez **Google Drive** et **Google Slides** et activez les différentes APIs.

Maintenant que les différentes APIs nécessaires sont configurées, il reste à configurer les authentifiants OAuth 2.0. Pour cela, il faut sélectionner l'écran de consentement OAuth, puis choisir Utilisateur externe et cliquez sur "créez". Enfin il faut remplir les informations requises.

Une fois cette étape effectuée, il faut créer l’ID client OAuth, il faut donc pour ”Type d’application” sélectionner ”Application web” et ensuite cliquer sur ”Créer”

Pour finir, il faut télécharger le fichier JSON des identifiants. C’est ce fichier qui permettra l’utilisation des APIs et fera le lien entre les futurs scripts python et google cloud platform.

8 Explication du Script Python

Maintenant que le problème de gestion des droits sur la manipulation des APIs google a été traité, il est temps de se focaliser sur l’utilisation de celle-ci via des scripts python mais aussi de la récupération des informations provenant des différents objets Salesforce.

Tout d’abord présentons l’architecture de l’ensemble des fichiers afin de pouvoir reproduire un environnement test de base:

Nous avons à la racine le dossier Salesforce, lui-même composé de trois sous-dossier : - ComposantRFP, Diapo et LinkSalesforce.

L’ensemble de ces dossiers ainsi que leur contenu sont disponible sur le git suivant :

[Lien du git](#)

Passons maintenant à l’architecture du code ainsi qu’à l’explications de chacune des différentes fonctions.

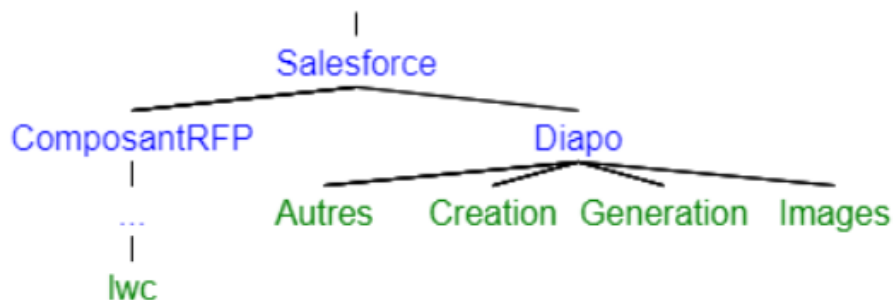


Figure 4:
Tree

Voici les différents fichiers présents dans le dossier lwc :

GoogleTestComponent.html : Définit l’interface utilisateur du composant

Web Lightning pour gérer les RFP dans Salesforce

GoogleTestComponent.js : Contient la logique Javascript permettant de gérer le composant.

GoogleTestComponent.js-meta.xml : Fichier de configuration décrivant les métadonnées du composant, indiquant où et comment le composant peut être utilisé dans Salesforce.

Voici les différents fichiers présents dans le dossier Salesforce/Diapo/Creation:

CreeZoneTxtSommaire :

recupère en entrée du code l'id de la présentation source ainsi que l'id de la diapo source, puis génère les zones de texte de la slide correspondant au sommaire

CreeZoneTxtExecSummary : Récupère les données Salesforce et les formate en texte puis crée des zones de texte et les insère dans la présentation google slides en les positionnant correctement.

CreeZoneTxtPricing : Récupère les données Salesforce et les formate en texte puis crée des zones de texte et les insère dans la présentation google slides en les positionnant correctement.

CreeZoneTxtDeclarationsConditionnelles : récupère en entrée du code l'id de la présentation source ainsi que l'id de la diapo source, puis génère les zones de texte de la slide correspondant au sommaire

CreeZoneTxtSolutions : Récupère les données Salesforce et les formate en texte puis crée des zones de texte et les insère dans la présentation google slides en les positionnant correctement.

Voici les différents fichiers présents dans le dossier Salesforce/Diapo/Génération:

GenereSlideDeclarationsConditionnelles : récupère en entrée du code l'id de la présentation source ainsi que l'id de la diapo source, puis génère la slide liées déclarations conditionnelles

GenereSlideExecSummary : Recupère les données Salesforce et les formate en texte puis crée des zones de texte et les insère dans la présentations google Slides en les positionnant correctement.

GenereSlidePricing : récupère en entrée du code l'id de la présentation source ainsi que l'id de la diapo source, puis génère la slide liée au pricing

GenereSlideSolution Recupère les données Salesforce liée au solutions Salesforce à proposer pour l'appel d'offre, les formate en texte puis crée des zones de texte et les insère dans la présentations google Slides en les positionnant correctement.

GenereSlidesSommaire : Genere la slide liée au sommaire en partant d'un squelette d'un sommaire de base.

GenereTemplate : S'occupe à partir d'une slide de générer une autre slide avec un id généré aléatoirement mais reprenant tous les éléments de la slide de base (sauf les templates qui ne sont pas des éléments cliquables).

PresentationRFP : Rassemble toutes les fonctions Genere... pour essayer de sortir une présentation correcte d'un appel d'offre.

Voici les différents fichiers présents dans le dossier Salesforce/Diapo/images:

Commerce Cloud
Marketing Cloud
Mulesoft
Sales Cloud
Salesforce Einstein
Service Cloud
Slack
Tableau
DeclarationConditionnelle

Voici les différents fichiers présents dans le dossier Salesforce/Diapo/Autre:

partageDrive : Permet de donner l'accès d'une présentation à une adresse mail

Une documentation plus détaillée contenant la raison d'être de chaque fonction est précisée dans le git et plus précisément dans le fichier texte "documentation détaillée".

9 Création du Composant RFP

La création du composant peut être découpée en deux types de composants. Tout d'abord si on souhaite uniquement générer les informations qui vont bien dans les tables, on s'intéressera à un composant constitué des 3 codes suivant :

`googleTestComponent.js` / `googleTestComponent.html` / `googleTestComponent.xml`

(On les retrouve dans le dossier lws de ComposantRFP dans le git donné ci-dessus)

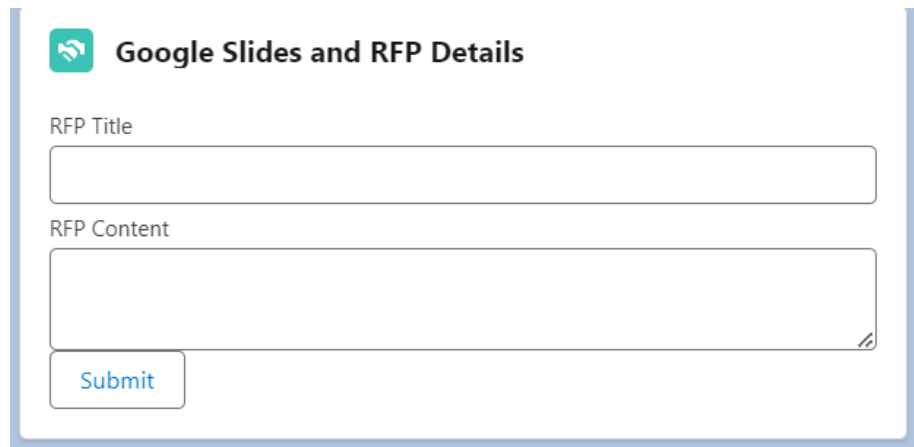
The image shows a screenshot of a Google Slides presentation slide. The slide has a title 'Google Slides and RFP Details' with a green icon to the left. Below the title, there are two text input fields. The first is labeled 'RFP Title' and the second is labeled 'RFP Content'. Below the 'RFP Content' field, there is a blue 'Submit' button. The entire form is enclosed in a light blue border.

Figure 5:
Composant RFP

Pour que ce composant soit fonctionnel, il faudra l'accompagner d'une classe Apex qui à la manière d'un flow s'occupera de créer ou modifier un record de la classe RFP. Je mets ci-dessous la classe à créer pour que ce soit fonctionnel :

Après avoir réalisé ce composant, on peut le modifier via la modification des trois codes précédents et le rajout de classe Apex qui requêtent un serveur heroku pour pouvoir automatiser l'exécution des script python. Je n'ai pas particulièrement avancé cette partie et c'est une des voie d'amélioration envisageable pour la suite de ce projet.

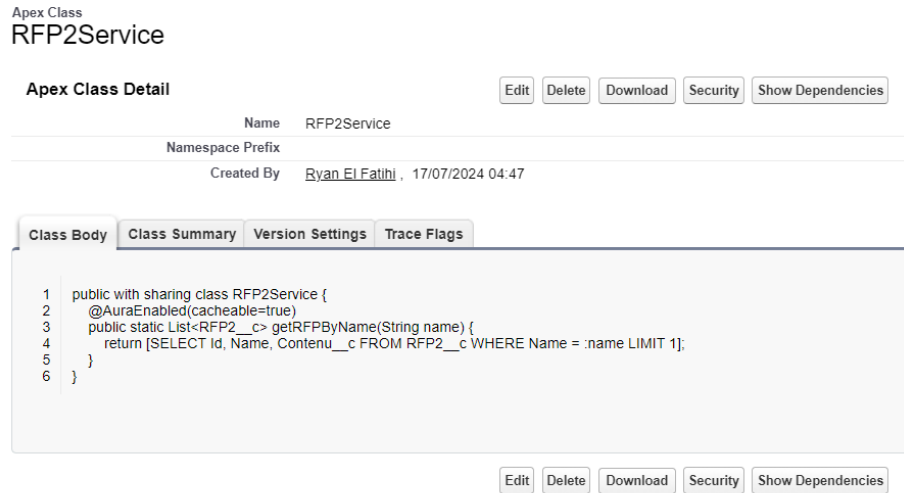


Figure 6:
Classe Composant RFP

10 Annexe

Limites du projet :

Ce projet contient quelques limites notables, parmi elles on pourrait citer les suivantes :

Le Rag n'a pas été utilisé finalement, il pourrait permettre d'atteindre une autre dimension sur la génération de texte et de choix pertinent vis à vis de l'appel d'offre étudié.

Tous les scripts ont été écrits en python, il existe une possibilité de les lancer de manière automatique depuis salesforce via une classe Apex requêtant un serveur heroku. Cependant j'ai appris il y a peu que tous coder en javascript pouvait être une alternative intéressante car on pourrait alors s'éviter le passage via un serveur heroku.

Les codes ne sont pas assez exhaustive et on pourrait grandement améliorer les squelettes de base, les prompts dans salesforce ainsi que le nombre de slides proposées.

Sources :

[Talent booster club - Data + IA + Focus RAG](#)

D'autres documents sur le RAG sont présent sur le git dont je rappelle le lien :

[Lien du git](#)