Divers

Quels sont les autres ressources disponibles avec Mail RP ?

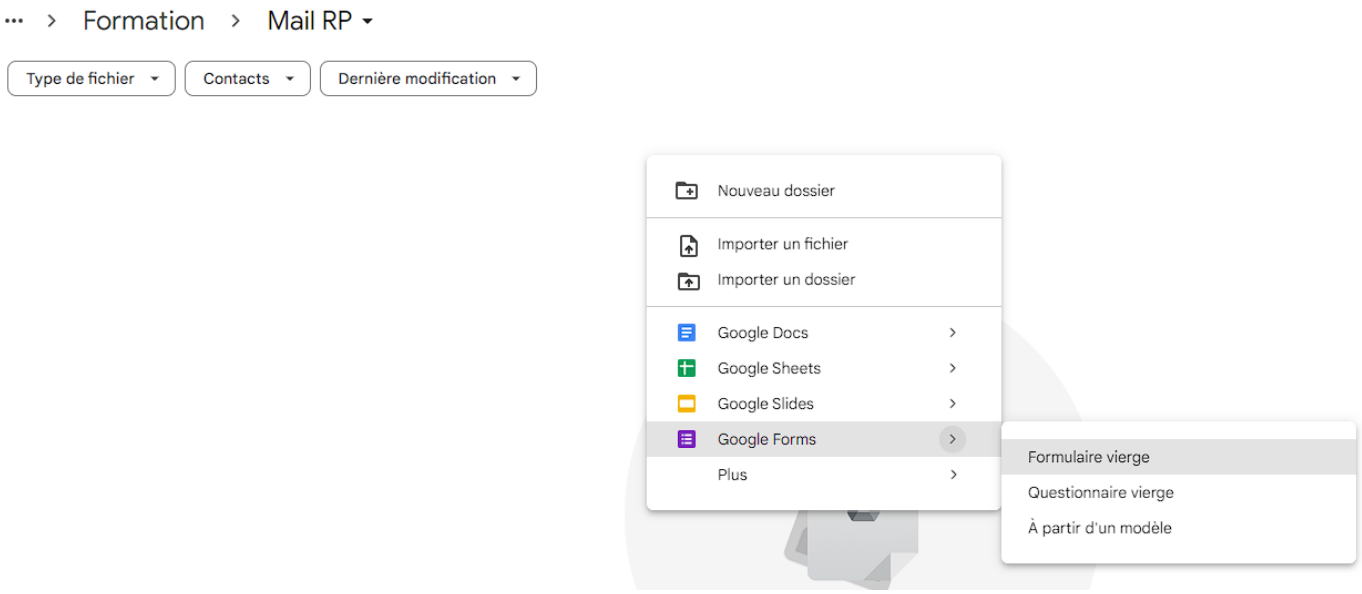
- Envoyer un mail à la soumission d'un Google Form
- Ressources FiveM
    - Utilisez la tablette XelBob-tab
    - Ressources GTA V - Installation
    - Ressources GTA V - Méthodes

Envoyer un mail à la soumission d'un Google Form

> **Avant propos :** Dans ce tutoriel, nous allons vous montrer comment faire à partir d'un Google form pour envoyer un fichier pdf par mail via le système Mail RP. En fonction de vos besoins, vous pourrez réaliser uniquement les étapes qui permettent de répondre à votre besoin.

1 - Créer votre Google Form

La première étape consiste à créer votre Google Form. Pour cette démonstration, nous allons partir sur la réalisation d'un compte rendu d'intervention que l'on va envoyer à notre client.



Information prestataire

Description (facultative)

Prénom Nom *

Réponse courte

Adresse mail *

Réponse courte

Information Client

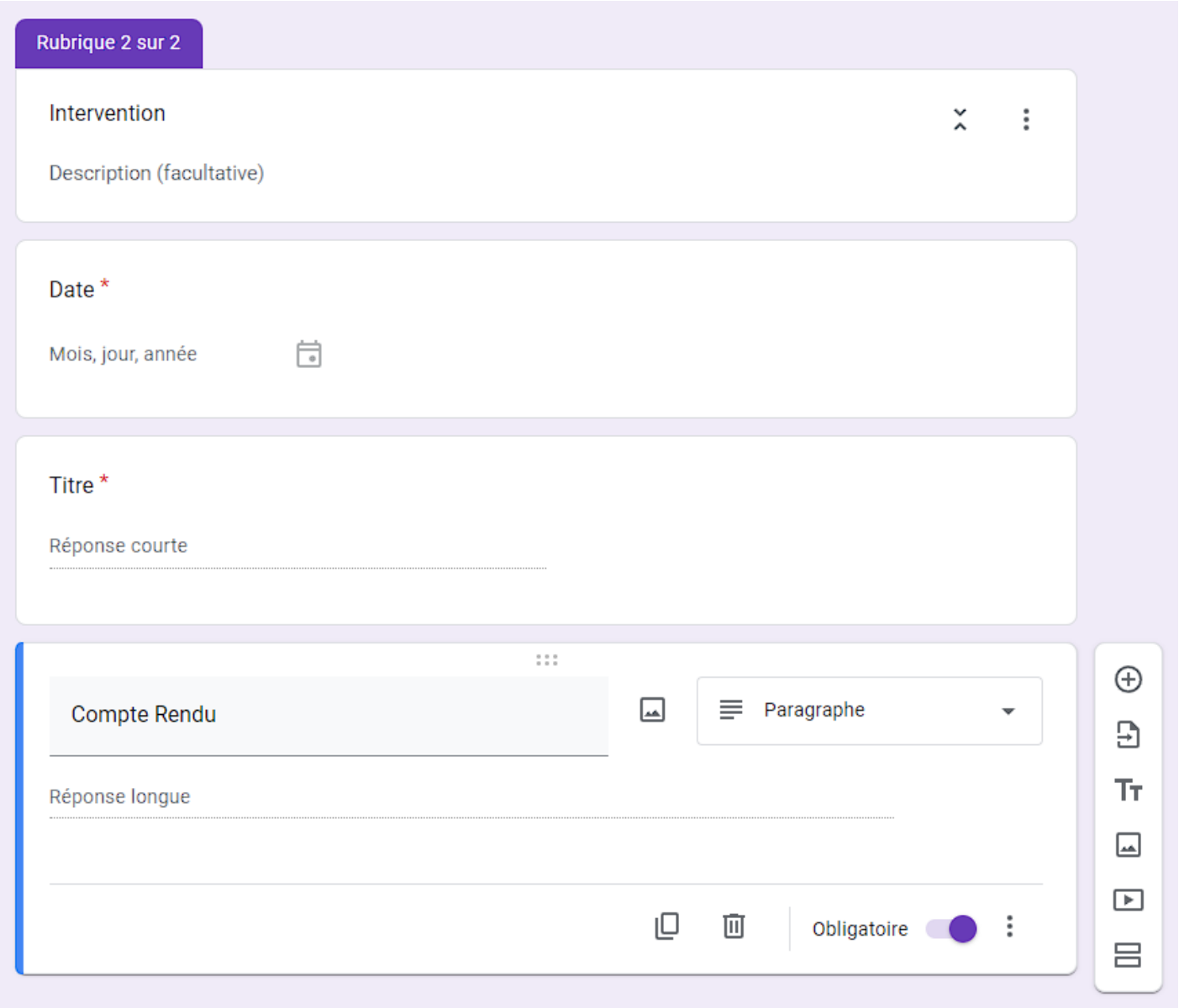
Description (facultative)

Prénom Nom *

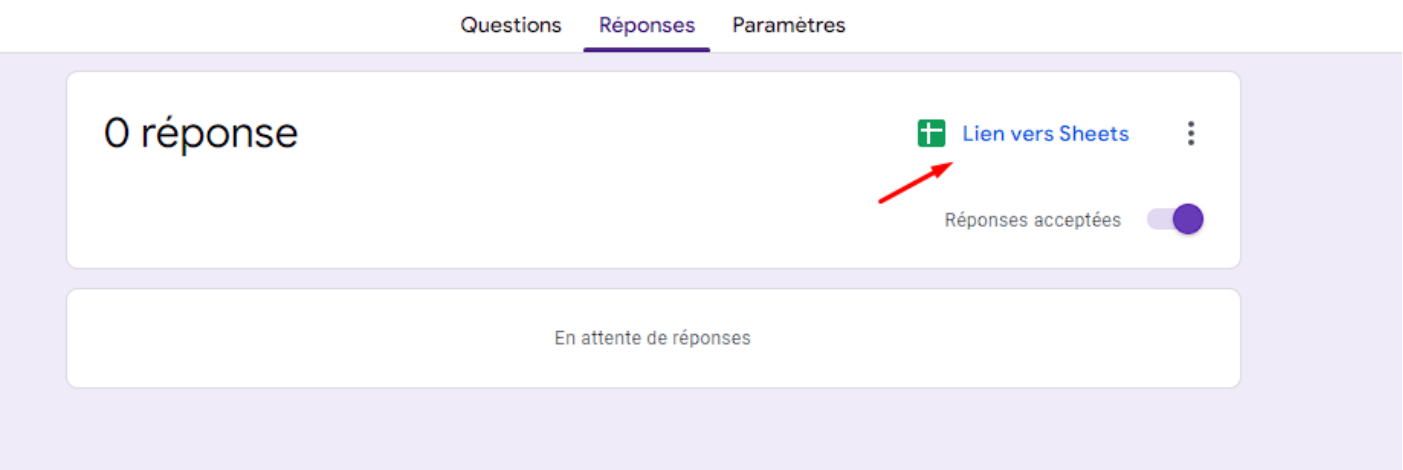
Réponse courte

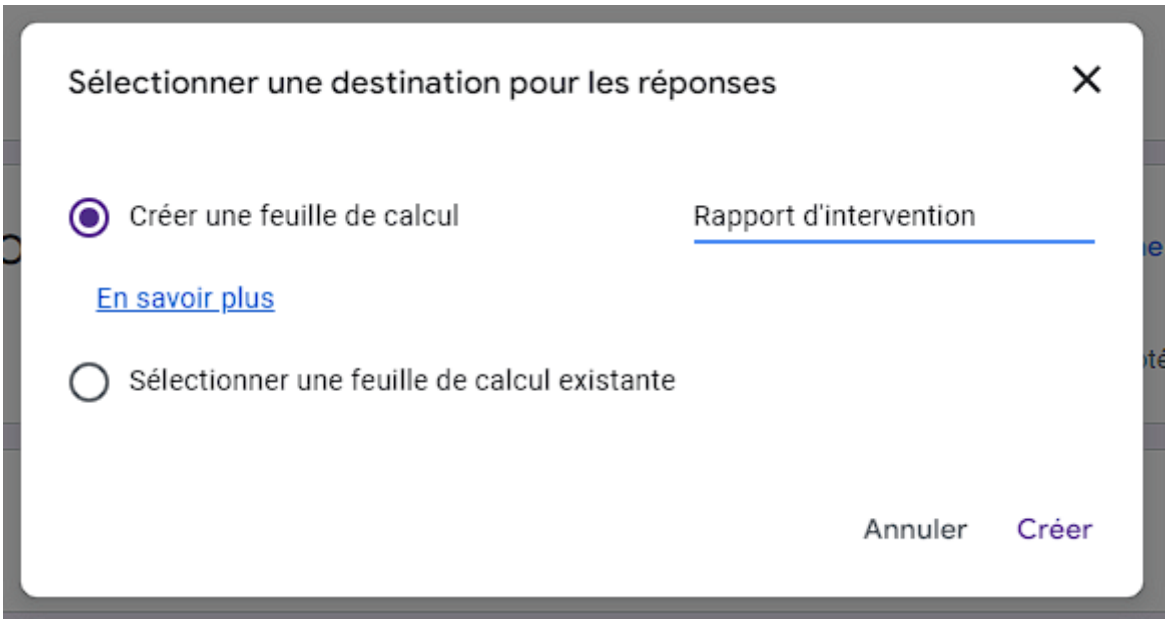
Adresse mail *

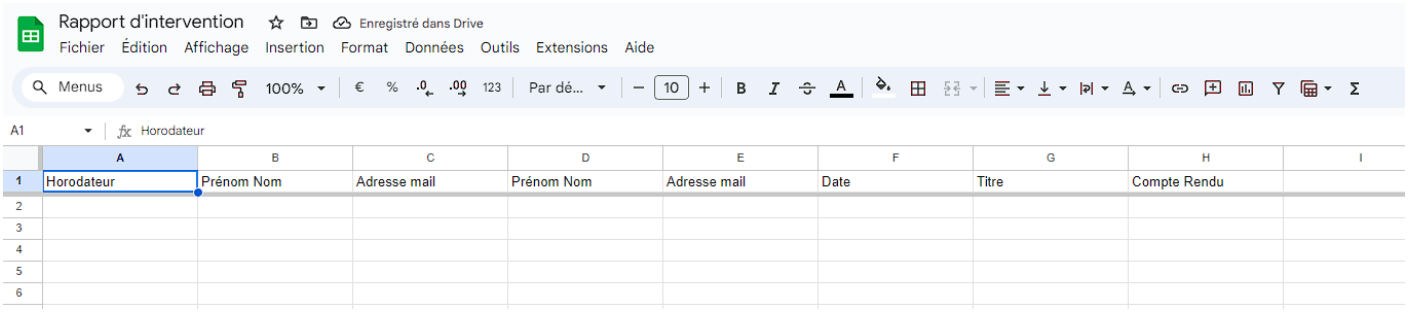
Réponse courte



Une fois votre formulaire créé, vous devez activer l'exportation des données vers un Google Sheet :

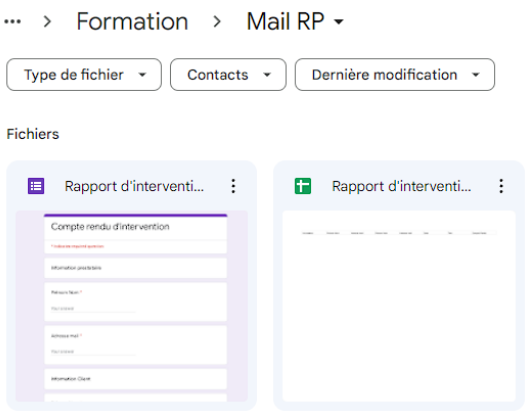


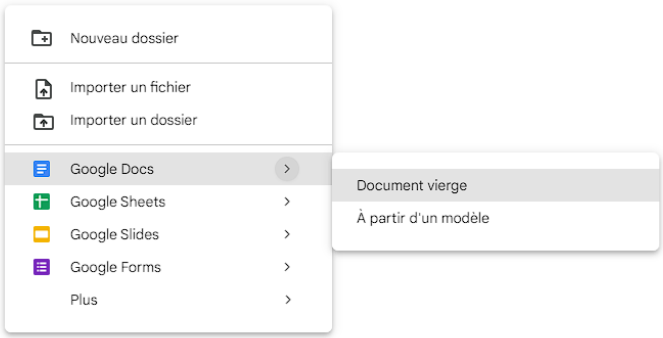




2 - Créer un template (GDoc)

Maintenant que le Google Form est créé, nous allons créer un template. Ce template réalisé sur Google Doc sera le modèle du rapport qui sera envoyé au client.





Dans le modèle, on retrouve plusieurs balises `{{name}}`. Ces balises vont nous permettre d'insérer des informations en les remplaçant par du contenu.

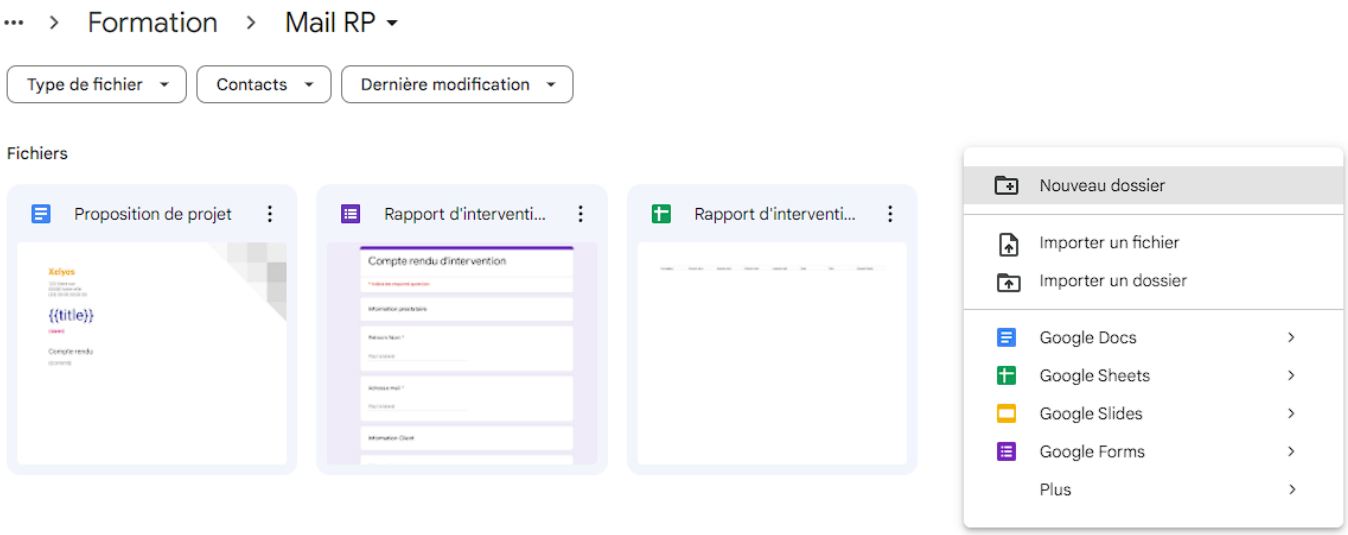
> Par exemple, **{{title}}** deviendra : **Audit de sécurité !**

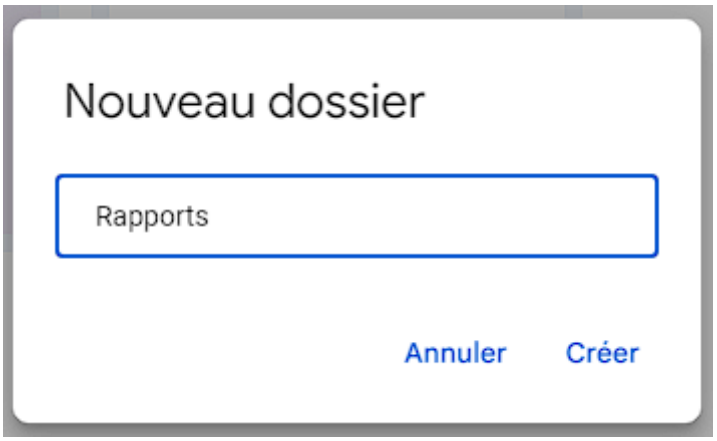
La nomenclature des balises est sans importance. Cependant, nous vous conseillons :

- d'essayer de mettre le nom de vos variables en : *camelCase* / *PascalCase*
- de ne pas mettre d'espace dans le nom
- de mettre un nom clair et facile à comprendre
- d'éviter de mettre des séparateurs dans le nom, comme : `.`, `-`, `/`, etc.

3 - Créer un répertoire de stockage

Lorsque nous allons générer les PDF, il faudra les stocker. C'est pourquoi nous allons créer un répertoire de stockage au sein de notre projet.





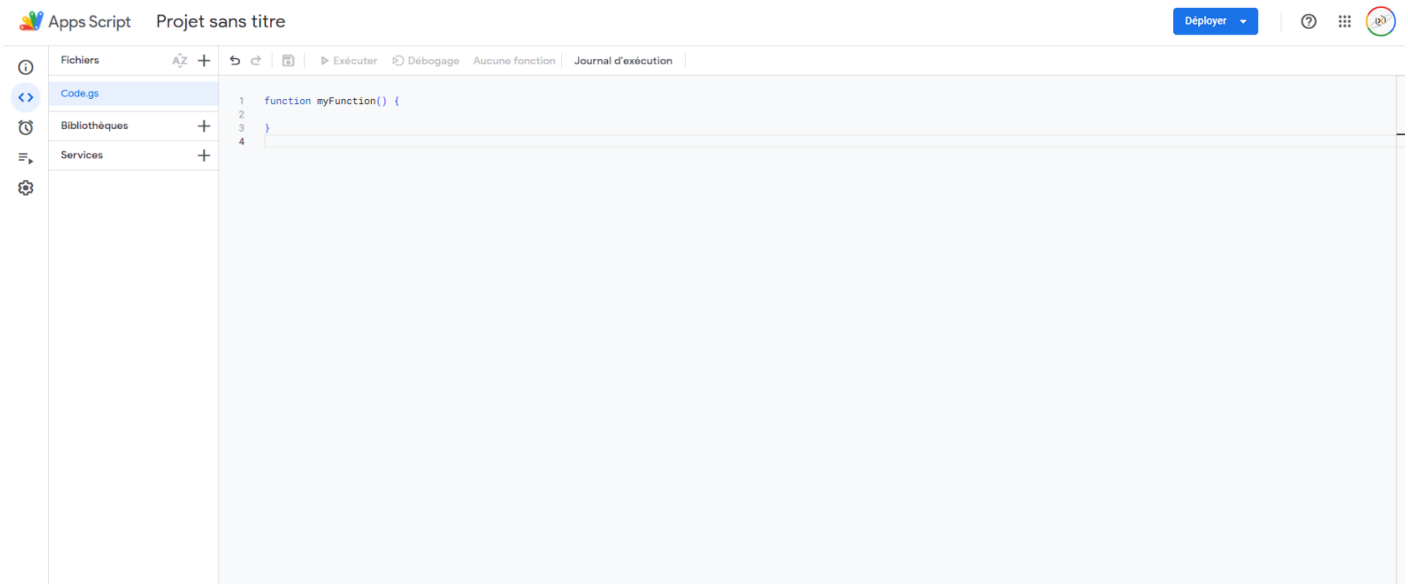
4 - Créer le script d'envoi

> Plus d'informations sur Apps Script : https://developers.google.com/apps-script/overview

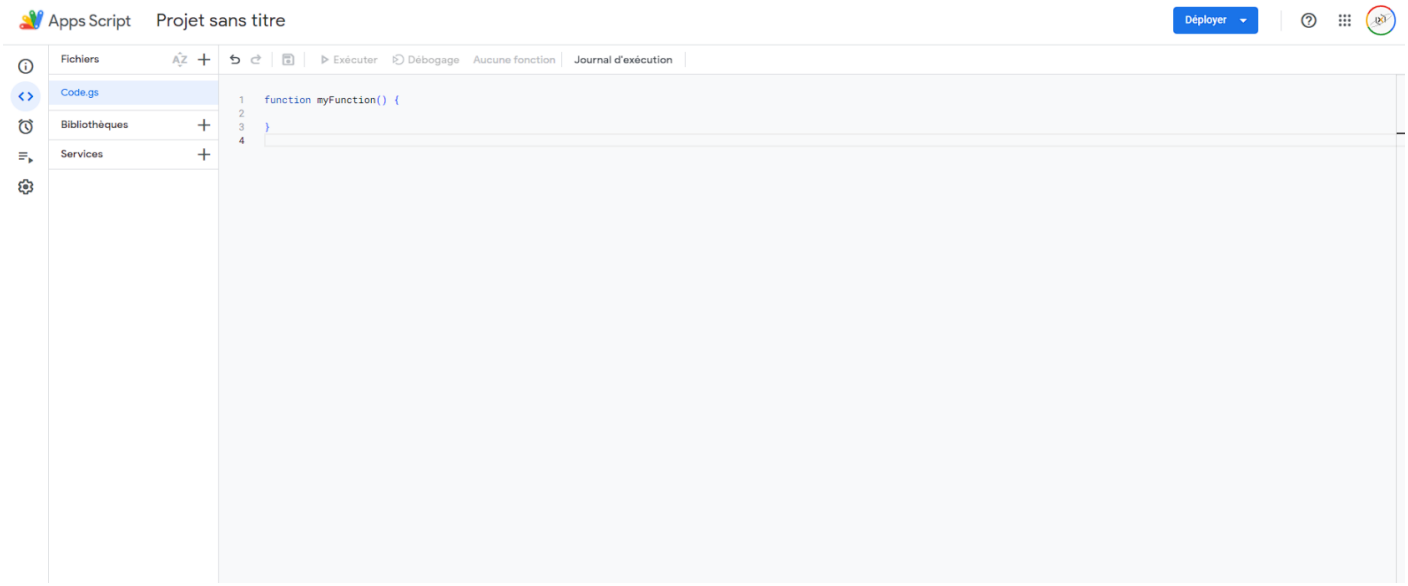
Maintenant que nous avons tous nos éléments prêts, nous allons créer le script permettant d'envoyer les réponses de notre formulaire.

Étape 1 : Initialisation du script

Sur le Google Sheet, nous allons ajouter un script. Pour cela, dans la barre extensions, sélectionnez Apps Script :



Voici à quoi ressemble notre projet par défaut :

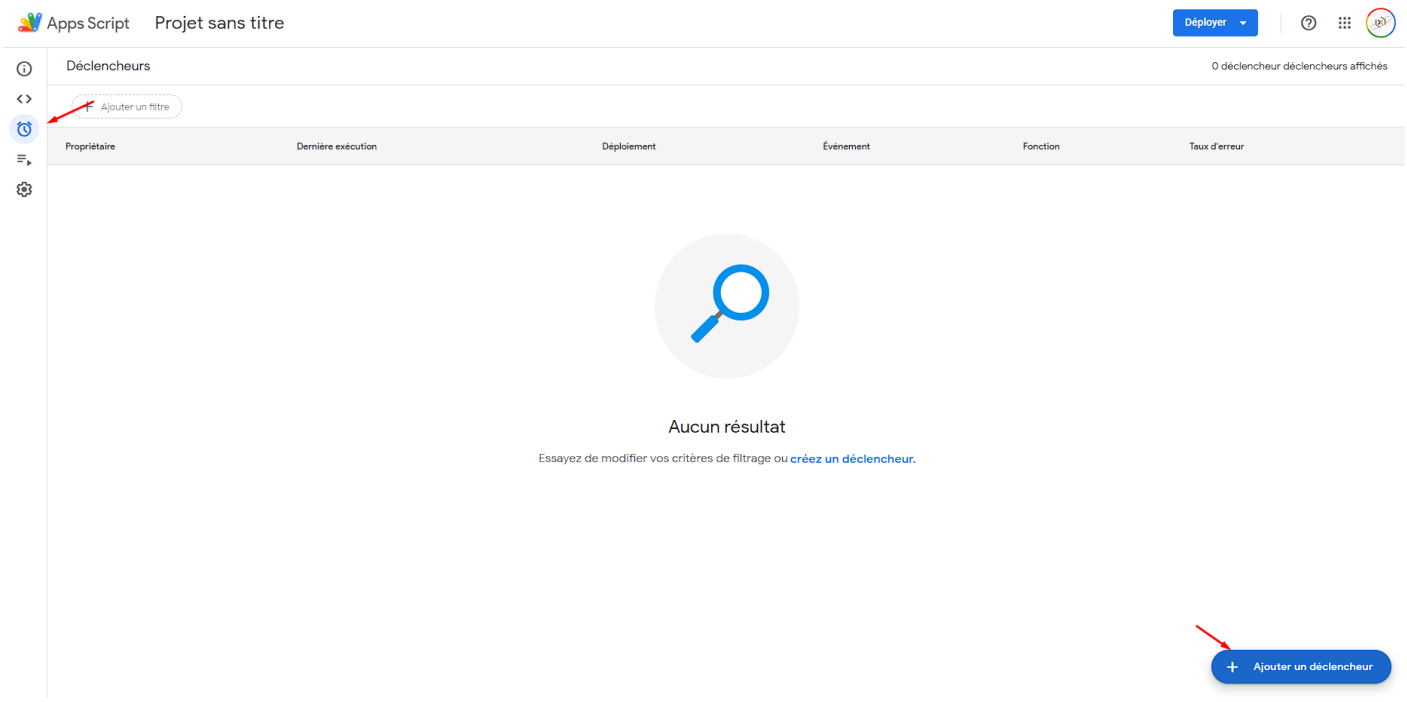


La première chose que l'on va faire c'est donner un nouveau nom à notre fonction et lui passer un paramètre `element` :

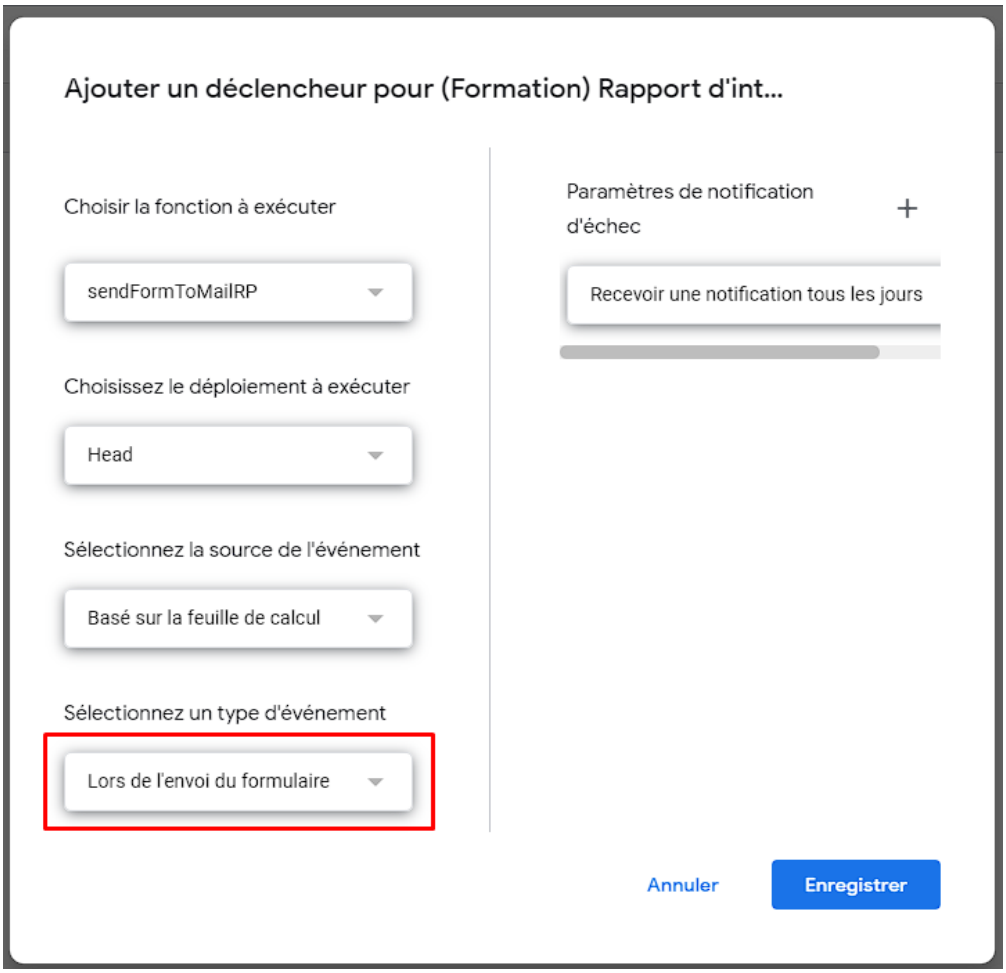
```
function sendFormToMailRP(element) {


}
```

Étape 2 : Ajouter un déclencheur pour l'envoi d'un formulaire

Pour que notre script fonctionne, on doit être en mesure de détecter l'ajout de donnée dans le Google Sheet. On va créer un déclencheur :



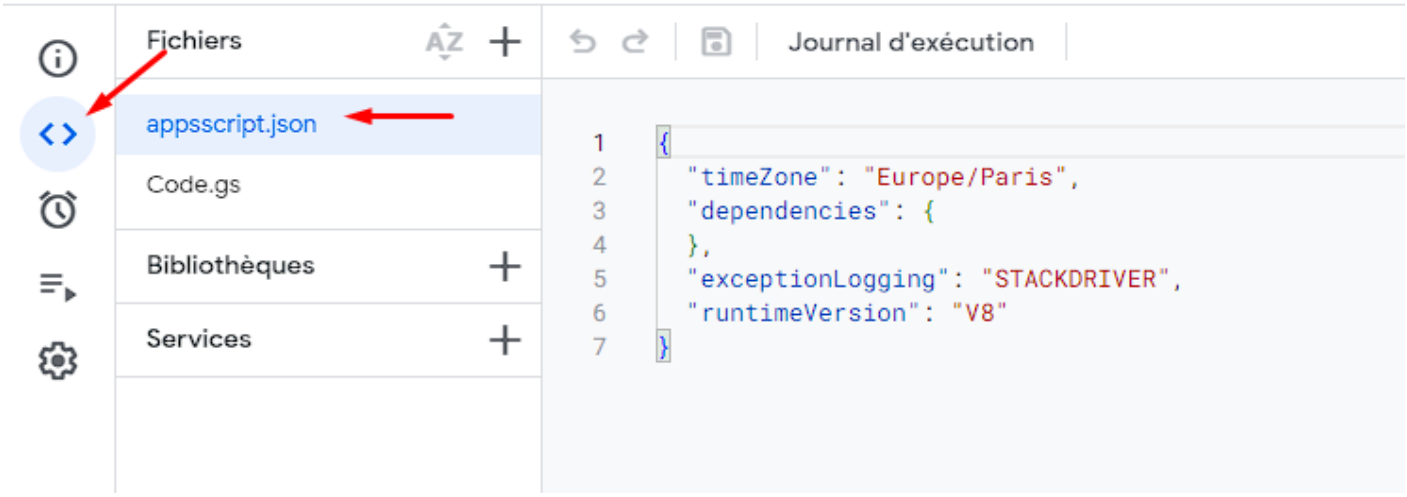
Voici les paramètres de notre déclencheur, attention à bien définir comme type d'évènement **Lors de l'envoi du formulaire** :



Étape 3 : Ajouter les dépendances nécessaires au fonctionnement du projet

Dans les paramètres du projet, afficher le manifeste `appscript.json` :

Retourner dans Éditeur, puis dans `appsscript.json` :



Remplacer le contenu du fichier par :

```
{
  "timeZone": "Europe/Paris",
  "dependencies": {
  },
  "oauthScopes": [
    "https://www.googleapis.com/auth/drive",
    "https://www.googleapis.com/auth/script.external_request",
    "https://www.googleapis.com/auth/documents"
  ],
```

```
  "exceptionLogging": "STACKDRIVER",
  "runtimeVersion": "V8"
}
```

Étape 4 : Récupérer les informations transmises par le formulaire

Pour récupérer les informations envoyées par le formulaire, nous allons utiliser la variable `element.values`. Cette variable est un tableau dont la structure des résultats est celle présente dans notre Google Sheet. On aura donc :

- **element.values[0]** : L'information Horodateur (quand a été envoyé le formulaire)
- **element.values[1]** : Prénom et Nom du prestataire
- **element.values[2]** : Adresse mail du prestataire
- **element.values[6]** : Titre de notre rapport
- **element.values[7]** : Contenu de notre compte rendu

La syntaxe dans notre script est donc :

```
function sendFormToMailRP(element) {
  var timestamp = element.values[0];
  ...
  var content = element.values[7];
}
```

Étape 5 : Récupérer un identifiant

Pendant toute la durée de notre script, nous aurons besoin de récupérer des identifiants (template, dossier,etc.). Pour récupérer l'identifiant d'un élement, il suffit d'aller récupérer l'information dans l'URL de celui-ci.

> Par exemple, si l'url de notre template est : `https://docs.google.com/document/d/ugIXX7Nv02TyBv3cbASf/edit`

> Alors l'identifiant de notre template sera : `ugIXX7Nv02TyBv3cbASf`.

Étape 6 : Remplissage de notre template

Nous allons récupérer l'id de notre template et de notre dossier de stockage. Ensuite, nous allons créer une copie de notre template dans le dossier de stockage et nous injecterons les résultats du formulaire dans ce template copié.

```
// Récupération du template par son ID
var template = DriveApp.getFileById("ugIXX7Nv02TyBv3cbASf");
```

```
// Récupération du répertoire de stockage par son id
var storage = DriveApp.getFolderById("3hK1OrcdRliKsV2R2HlN");

// Création d'un nouveau document dans le répertoire de stockage
var fileName = "Rapport d'intervention | " + timestamp + " | " + clientName; // Choisissez le
nom de fichier que vous voulez mettre
var newTemplate = template.makeCopy(fileName, storage);

// Ouverture du document
var document = DocumentApp.openById(newTemplate.getId());

// Injection des informations dans le template
var body = document.getBody();
body.replaceText("{{title}}", title);
...
body.replaceText("{{content}}", content);

// Sauvegarde des modifications
document.saveAndClose();
```

> Veuillez noter que la méthode `body.replaceText("{{title}}", title);` fait appel dans un premier temps à la balise définie dans votre template et dans un second temps au nom de la variable qui contient l'information récupéré en début de programme.

Étape 7 : Export du document en PDF

Maintenant que notre template est complété, nous allons le transformer en PDF :

```
// Export en PDF
var pdfBlob = document.getAs('application/pdf');
pdfBlob.setName(fileName + '.pdf');

var pdf = DriveApp.createFile(pdfBlob);
pdf.setSharing(DriveApp.Access.ANYONE_WITH_LINK, DriveApp.Permission.VIEW);
pdf = pdf.makeCopy(fileName, storage);

// Suppression du fichier doc
DriveApp.getFileById(document.getId()).setTrashed(true);
```

> Plus d'information sur les accès : Accès Google Script

> Plus d'information sur les permissions : Permission Google Script

Étape 8 : Envoyer le PDF par mail

On y est presque, notre PDF est maintenant dans notre espace de stockage. Il ne nous reste plus qu'à l'envoyer à notre destinataire.

> Pour cela, nous allons avoir besoin d'une clé d'accès à l'API Mail RP (Fonctionnement de l'API).

Dans le script, remplacez `api-key`, par votre clé API :

```
// Envoyer un mail avec Mail RP
UrlFetchApp.fetch('https://mail-rp.com/api/v1/api-key/mail/send', {
  'method': 'post',
  'contentType': 'application/json',
  'payload' : JSON.stringify({
    "from" : providerEmail,
    "to" : clientEmail,
    "subject": "Rapport d'intervention",
    "content": "<p>Bonjour " + clientName + ",</p><br>Vous trouverez ci-joint le rapport d'intervention du ...",
    "openConfirmation": false,
    "attachments": [
      {
        "name": pdf.getName() + '.pdf',
        "url": pdf.getDownloadUrl() // ou pdf.getUrl()
      }
    ]
  })
});
```

Champs	Description	Exemple
from	Email de l'expéditeur	Ici notre prestataire : `providerEmail`
to	Email du destinataire	Ici le client : `clientEmail`

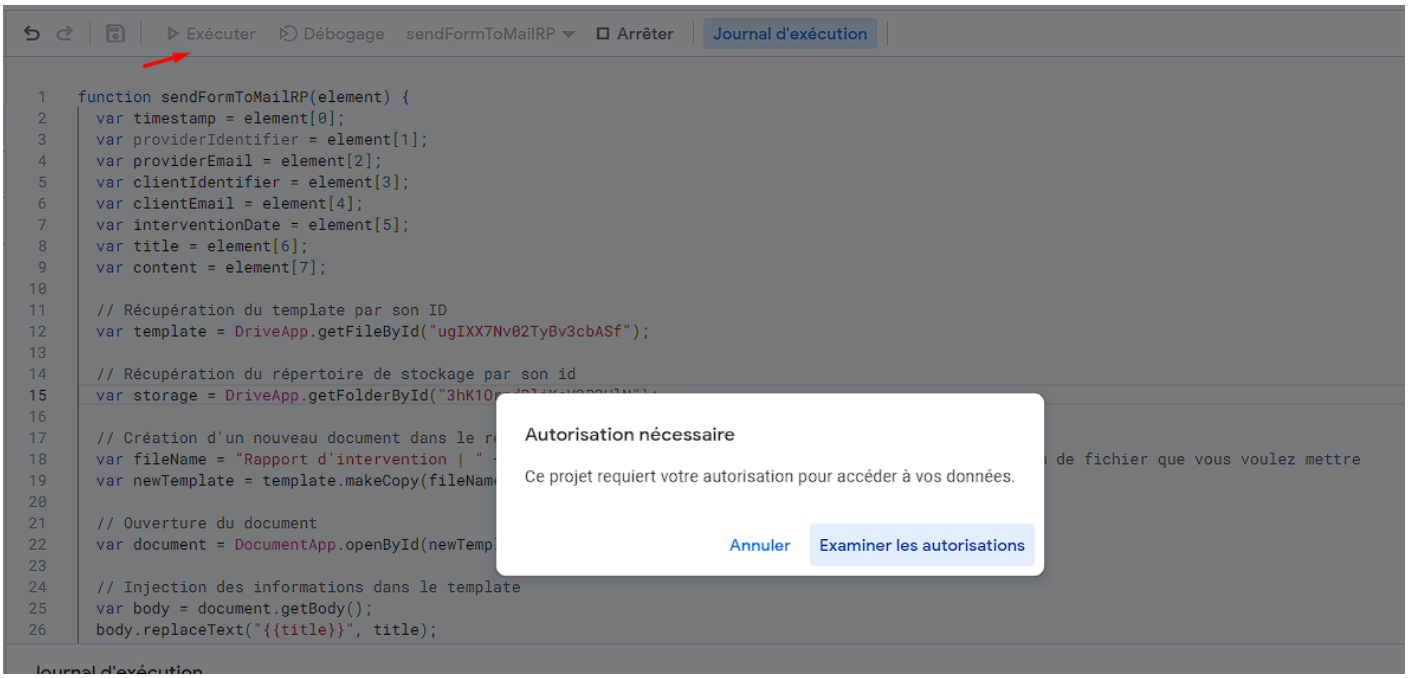
subject	Sujet du mail	Ex : `Rapport d'intervention`
content	Contenu du mail	Pour un meilleur résultat, le contenu du mail doit être écrit avec des balises HTML
openConfirmation	Recevoir une confirmation d'ouverture ?	true | false
attachements	Notre PDF attaché en pièce joint	Consulter la documentation de l'API pour plus d'information

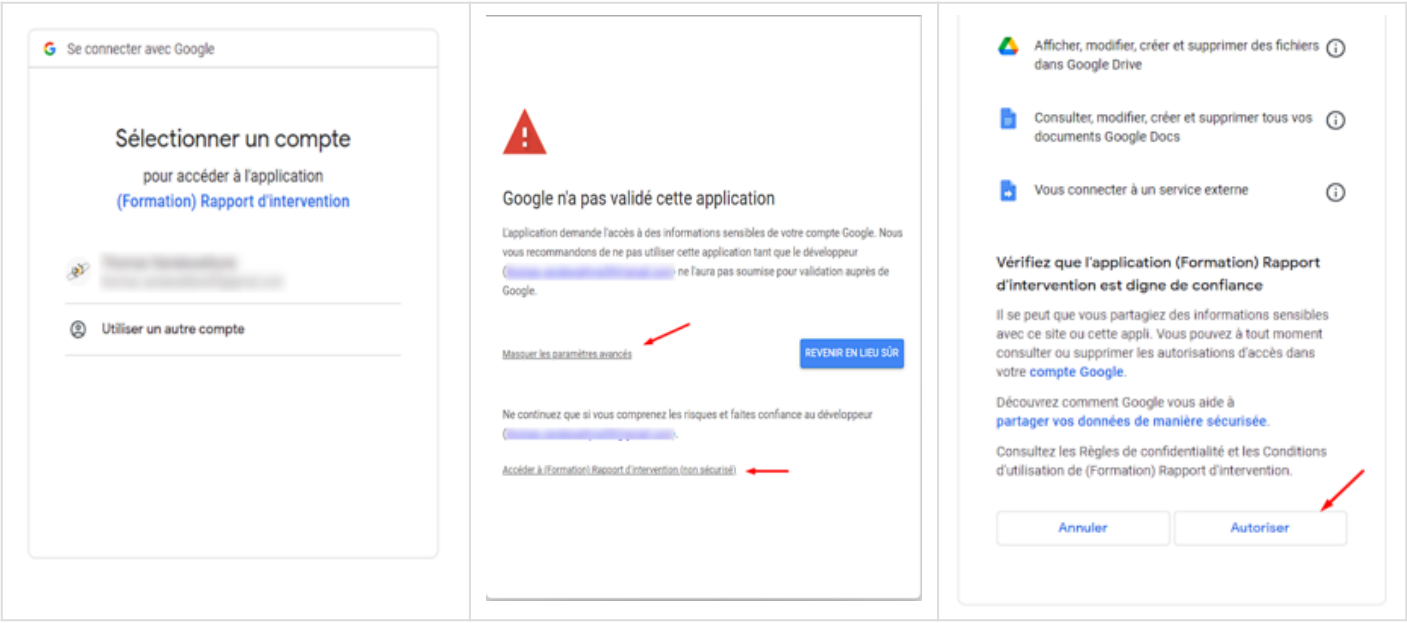
Étape 9 : Vérification des permissions

> Une fois que vous avez terminé votre script. Nous vous conseillons d'appuyer au moins une fois sur **Exécutee**.

Google va alors vous demander la permission d'accéder à certains éléments de votre compte, comme dans notre cas l'accès aux répertoires et documents.

> Lors de cette étape, assurez-vous d'être l'auteur du script et que personne d'autre n'y à accès afin de limiter les risques.





5 - Script Complet

```
function sendFormToMailRP(element) {
  var timestamp = element[0];
  ...
  var content = element[7];

  // Récupération du template par son ID
  var template = DriveApp.getFileById("ugIXX7Nv02TyBv3cbASf");

  // Récupération du répertoire de stockage par son id
  var storage = DriveApp.getFolderById("3hK1OrcdRliKsV2R2HlN");

  // Création d'un nouveau document dans le répertoire de stockage
  var fileName = "Rapport d'intervention | " + timestamp + " | " + clientName; // Choisissez le nom de fichier que vous voulez mettre
  var newTemplate = template.makeCopy(fileName, storage);

  // Ouverture du document
  var document = DocumentApp.openById(newTemplate.getId());

  // Injection des informations dans le template
  var body = document.getBody();
  body.replaceText("{{title}}", title);
  ...
  body.replaceText("{{content}}", content);
```

```
  // Sauvegarde des modifications
  document.saveAndClose();

  // Export en PDF
  var pdfBlob = document.getAs('application/pdf');
  pdfBlob.setName(fileName + '.pdf');

  var pdf = DriveApp.createFile(pdfBlob);
  pdf.setSharing(DriveApp.Access.ANYONE_WITH_LINK, DriveApp.Permission.VIEW);
  pdf = pdf.makeCopy(fileName, storage);

  // Suppression du fichier doc
  DriveApp.getFileById(document.getId()).setTrashed(true);

  // Envoyer un mail avec Mail RP
  UrlFetchApp.fetch('https://mail-rp.com/api/v1/api-key/mail/send', {
    'method': 'post',
    'contentType': 'application/json',
    'payload' : JSON.stringify({
      "from" : providerEmail,
      "to" : clientEmail,
      "subject": "Rapport d'intervention",
      "content": "<p>Bonjour " + clientName + ",</p><br>Vous trouverez ci-joint le rapport
d'intervention du ...",
      "openConfirmation": false,
      "attachments": [
        {
          "name": pdf.getName() + '.pdf',
          "url": pdf.getDownloadUrl() // ou pdf.getUrl()
        }
      ]
    })
  });
}
```

Ressources FiveM

Ressources FiveM

Utilisez la tablette XelBob-tab

XelBob-tab est le produit de l'association entre Xelyos et Bob's&Co. Cette tablette utilisable sur Fivem vous permet d'afficher le contenu de sites internet directement sur votre serveur.

Afficher plus d'info

Ressources GTA V - Installation

Cette ressources n'est plus fonctionnelle pour le moment !

Installer la ressource

Pour récupérer la clé vous permettant d'utiliser l'api de Mail RP, rendez vous sur la page :

Récupérer sa clé API

Voici deux méthodes pour installer la ressources sur votre serveur :

Avec git :

Rendez vous dans le dossier `resources` de votre serveur, et utiliser la commande suivante :

```
git clone https://github.com/XelyosTeam/fivem-resources.git [xelyos]
```

Un dossier va allors de créer sous le nom **[xelyos]**, vérifiez bien que dans ce répertoire vous trouverez le répertoire **mailrp**.

Sans git :

Rendez vous sur le git : **Last releases Git Xelyos Resources FiveM** et télécharger le fichier `.zip`. Déplacez le fichier zip téléchargé dans votre répertoire `resources`, et faîtes *extraire-ici*, renommer le nouveau fichier installer en **[xelyos]**

Activer la ressource

Lier à l'API :

Pour lié votre serveur FiveM avec l'api de Mail RP, vous devez vous rendre dans le fichier `resources/[xelyos]/mailrp/server.lua`, au niveau de la ligne 6, remplacez **apiToken** par votre clé.

Activer la ressource :

Pour activer la ressource sur votre serveur, ajouter dans votre `server.cfg` les lignes suivantes :

```
# Xelyos solutions
ensure mailrp
```

Si vous utilisez déjà des ressources Xelyos, vous n'avez qu'à ajouter la ligne à la suite des autres.

Utiliser la ressource

La fonction callback :

```
function requestAction(statusCode, resultData)
  -- Action qui suit la requête
end
```

Lorsque vous allez faire appel à l'API, un résultat est envoyé en fonction de la requête que vous utilisez. Pour exploiter cette réponse, vous devrez utilisez une fonction *callback*, dans cette fonction écrivez les actions qui devront suivre la requête. Vous pouvez renommer la fonction *callback* comme vous le voulez.

Variable	Définition
statusCode **(obligatoire)**	Résultat du status de la requête (200 : réussite | 403 : échec)
resultData **(obligatoire)**	Données retournées au format JSON

Appelez le code depuis un fichier serveur :

Pour utiliser l'api depuis un fichier serveur, vous devez utiliser la structure suivante :

```
function requestAction(statusCode, ResultData)
  ...
end

TriggerEvent('xelyos:MailRP', function(mailRP)
    mailRP:methode(requestAction)
end)
```

Afficher la liste des méthodes

Vous trouverez la liste des méthodes ci-dessous !

Appelez le code depuis un fichier client :

Pour utiliser l'api depuis un fichier serveur, vous devez utiliser la structure suivante :

```
function requestAction(statusCode, ResultData)
  ...
end

TriggerServerEvent('xelyos:MailRP', function(mailRP)
    mailRP:methode(requestAction)
end)
```

Afficher la liste des méthodes

Exemple d'utilisation - Récupérer le nom de son serveur

```
function showServerInfo(statusCode, resultData)
    print("Status", statusCode) -- Status de la requête
    print('Result', resultData) -- Informations retournées
  print('Server name', resultData.name) -- Récupération du nom du serveur
    print('Server count api request', resultData.api.count) -- Récupérer le nombre de requêtes
effectuées au total pour le serveur
end

AddEventHandler('onResourceStart', function(resourceName)
    if GetCurrentResourceName() == resourceName then
      -- Lors de l'initialisation de la ressource
        TriggerEvent('xelyos:MailRP', function(mailRP)
            mailRP:getServer(showServerInfo)
        end)
    end
end)
```

Ressources GTA V - Méthodes

Cette ressources n'est plus fonctionnelle pour le moment !

Chaque méthode possède dans ses paramètres une variable appelée `callBack`, cette variable correspond à la fonction que vous utilisez pour le traitement des données renvoyées par l'API. Chaque variable utilisée dans les différentes méthodes sont expliquées sur le document API.

---

getServer()

Utilité : Récupérer les informations sur son serveur

Doc API : Gestion du serveur - Informations générales

Appel :

```
mailRP:getServer(callBack)
```

---

findSubdomains()

Utilisé : Récupérer la liste des sous domaines de votre serveur

Doc API : Gestion des sous domaines - Liste des sous domaines

Appel :

```
mailRP:findSubdomains(callBack)
```

---

findSubdomain(name)

Utilisé : Récupérer les informations d'un sous domaine avec ses adresses mails

Doc API : Gestion des sous domaines - Détails d'un sous domaine

Appel :

```
mailRP:findSubdomain(callBack, name)
```

---

addSubdomain(name, public, manager, carnet)

Utilisé : Créer un nouveau sous domaine

Doc API : Gestion des sous domaines - Ajouter un sous domaine

Appel :

```
mailRP:addSubdomain(callBack, name, public, manager, carnet)
```

---

findAddress(email)

Utilisé : Récupérer les informations d'une adresse mail

Doc API : Gestion des adresses - Récupérer une adresse mail

Appel :

```
mailRP:findAddress(callBack, email)
```

---

addAddress(name, firstname, email, password)

Utilisé : Créer une adresse mail

Doc API : Gestion des adresses - Créer une adresse mail

Appel :

```
mailRP:addAddress(callBack, name, firstname, email, password)
```

sendEmail(from, to, subject, content, confirmOpened, attachments)

Utilisé : Envoyer un mail depuis une adresses à un ou plusieurs destinataires

Doc API : Gestion des mails - Envoyer un mail

Appel :

```
mailRP:sendEmail(callBack, from, to, subject, content, confirmOpened, attachments)
```