

Contexte

L'outil de création de courriers PDF a été développé en lien avec notre système de gestion électronique des documents (GED) EzGED, développé par EzDev. Cette solution a été mise en place pour répondre à un besoin spécifique dans le cadre de la gestion dématérialisée des courriers au sein des mairies.

De nombreux échanges oraux ont lieu entre les agents municipaux et leurs interlocuteurs, mais la GED était initialement axée sur la gestion des courriers physiques ou par mail. Par conséquent, il était nécessaire de trouver une solution permettant de conserver une trace écrite de ces échanges oraux, même en l'absence de courrier associé.

Afin de compléter les fonctionnalités existantes, j'ai développé cet outil de création de courriers PDF. Son objectif est de permettre aux agents municipaux de générer des courriers à partir d'une page web et de les intégrer facilement dans la GED existante. Le tout devant être responsive car une version mobile de EzGED existe.

Intégrer cet outil à la GED, améliorera sa capacité à gérer les échanges oraux. Cette synergie entre l'outil de création de courriers PDF et la GED EzGED renforce la traçabilité des décisions prises lors des entretiens téléphoniques ou des discussions, contribuant ainsi à une gestion plus complète et efficace des flux de courriers.

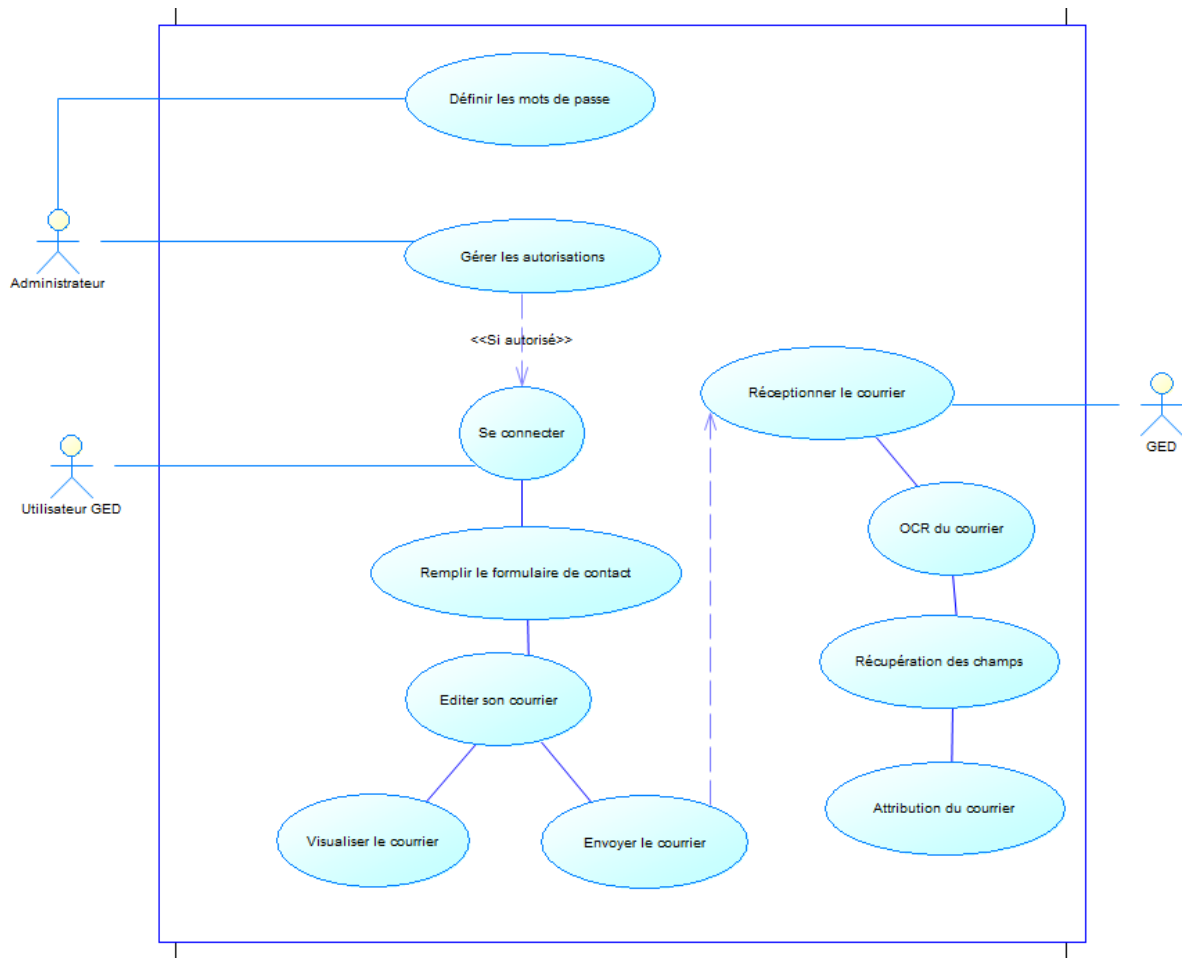
En conclusion, l'outil de création de courriers PDF représente une solution innovante pour répondre au besoin de traçabilité des échanges oraux dans le cadre de la gestion dématérialisée des courriers. Cette synergie renforce la transparence, la traçabilité et la qualité du service rendu par les mairies, tout en facilitant la recherche et la consultation ultérieure des informations.

Fonctionnement

Le but est ici de présenter le déroulement de l'utilisation de l'outil. On part du principe que l'utilisateur est déjà connecté à la GED et qu'il sait comment accéder à la page pour se connecter à l'outil.

Pour illustrer le déroulement avec l'implications des différents acteurs voici le diagramme suivant :

Chaque cas est détaillé d'avantages sous le diagramme



L'administrateur est chargé de gérer qui a le droit d'utiliser l'outil, il a donc la tâche de définir les mots de passe via autorisation.php et peut également au cas par cas cocher ou non pour changer les autorisations même après la création du compte d'accès à l'outil

L'utilisateur doit donc se connecter à l'outil depuis la GED, si il est autorisé et qu'il a les bons identifiants il accède à l'outil de rédaction

Dans l'ordre logique des choses il remplit le formulaire pour pouvoir faciliter le classement du courrier par la suite.

Il rédige ensuite son courrier en utilisant toutes les fonctionnalités mises à sa disposition.

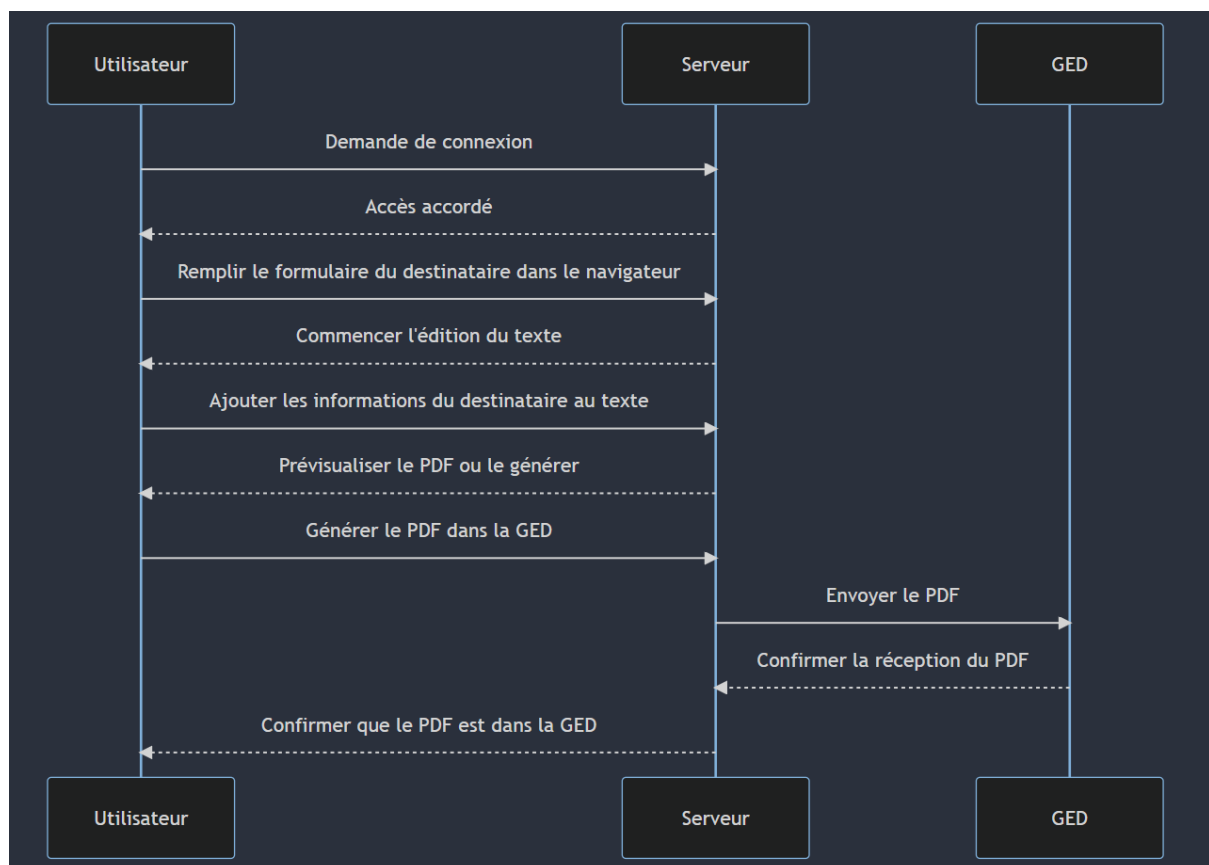
Il peut vérifier le résultat final de son pdf à l'aide du bouton **visualiser** et ensuite envoyer définitivement son courrier dans la GED en cliquant sur **générer**

La GED va donc réceptionner le courrier

Elle se chargera ensuite de faire un OCR complet du document pour pouvoir remplir automatiquement les champs.

A l'aide de ses champs elle sera par la suite comment classer et attribuer le courrier

Également un diagramme de séquence de l'usage classique d'un utilisateur de la connexion à l'envoi du pdf dans la GED :



Tables utilisés

Présentation et explications des tables utilisées.

Pour plus de compréhension je présente ici uniquement les champs utilisés dans l'application

courriers_specif_users

secusr_id (int)

password (texte)

autorisation (bool)

Cette table permet l'accès à mon outil et est notamment rempli par la page autorisation.php

Le champ **autorisation** est un booléen qui est sur true dans le cas ou l'utilisateur concerné est autorisé à accéder à l'application de création de courriers.

Il y a une autorisation qui est mise en place car cet outil peut générer plusieurs pdf et donc rajouter une charge de travail pour le moteur de la GED. Utiliser la création de pdf via cet outil doit donc être réglementé et pas accessible à tous les utilisateurs pour éviter d'éventuelles erreurs.

Il est possible d'imaginer une formation à cet outil et une fois la formation validé l'activation du champ autorisation.

Le **secusr_id** permet de récupérer le nom d'utilisateur contenu dans la table _secusr pour pouvoir récupérer notamment son login afin de savoir de qui il s'agit.

Le **password** est un mot de passe supplémentaire en plus de celui pour accéder à la ged il est défini par l'administrateur via la page autorisation.php.

Il est utilisé par l'utilisateur au niveau du formulaire de connexion pour accéder à la création de pdf.

courriers

objet (texte)

tel_specif (int)

mail_specif (texte)

destinataire_specif (texte)

expediteur_specif (texte)

chrono_ezged (int)

full_txt (fulltexte)

Cette table n'est pas utilisée par mon application mais elle est utilisée par la ged notamment au niveau de la reconnaissance automatique des champs.

En effet la GED va venir analysé mon pdf et à l'aide d'expressions régulières récupérer les champs contenus dans le pdf :

Expediteur : CAP

Destinataire : mc.accueil

Email : test@mail.fr

N° Tel : 0707070707

Objet : test

Le champ "**expediteur_specif**" indique l'expéditeur du courrier, qui est "CAP" dans ce cas précis.

Le champ "**destinataire_specif**" indique le destinataire du courrier, qui est "mc.accueil".

Le champ "**mail_specif**" indique l'adresse e-mail associée à ce courrier, qui est "test@mail.fr".

Le champ "**tel_specif**" indique le numéro de téléphone associé à ce courrier, qui est "0707070707".

Le champ "**objet**" indique le sujet ou l'objet du courrier, qui est "test".

Le champ **chrono_ezged** est un champ qui se remplit automatiquement qui sert de clé unique pour retrouver un courrier dans la GED, il sera rempli dans notre cas lorsque l'utilisateur enverra un courrier depuis notre outil dans la GED.

full_txt est lui aussi rempli automatiquement il contient pour chaque courriers tous les mots que l'OCR (Reconnaissance optique de caractères) a pu récupérer sur le PDF.

Il servira pour pouvoir rechercher facilement un courrier dans la GED.

_secusr

id (int)

login (texte)

password (texte)

super (bool)

lock (bool)

mail (texte)

Cette table contient toutes les informations sur les utilisateurs qui ont un compte dans la ged

Id est la clé primaire unique

le **login** est le nom avec lequel l'utilisateur se connecte, pour nous : demo_celio

Le **password** est son mot de passe (pour accéder à la GED pas pour accéder à la création de PDF)

super permet de définir si l'utilisateur est un administrateur

Lock permet de définir si l'utilisateur est verrouillé ou non, un trigger le passe sur true automatiquement si l'utilisateur se trompe 5 fois d'affilée (pour éviter les tentatives d'intrusions par exemple). Il est aussi utilisé dans le cas où des utilisateurs ne se servent pas correctement de la GED pour limiter la casse.

Mail correspond au mail de l'utilisateur et permet de lui envoyer des notifications dans le cas où il a des actions à effectuer sur la GED.

cap_contact

id (int)

lib (texte)

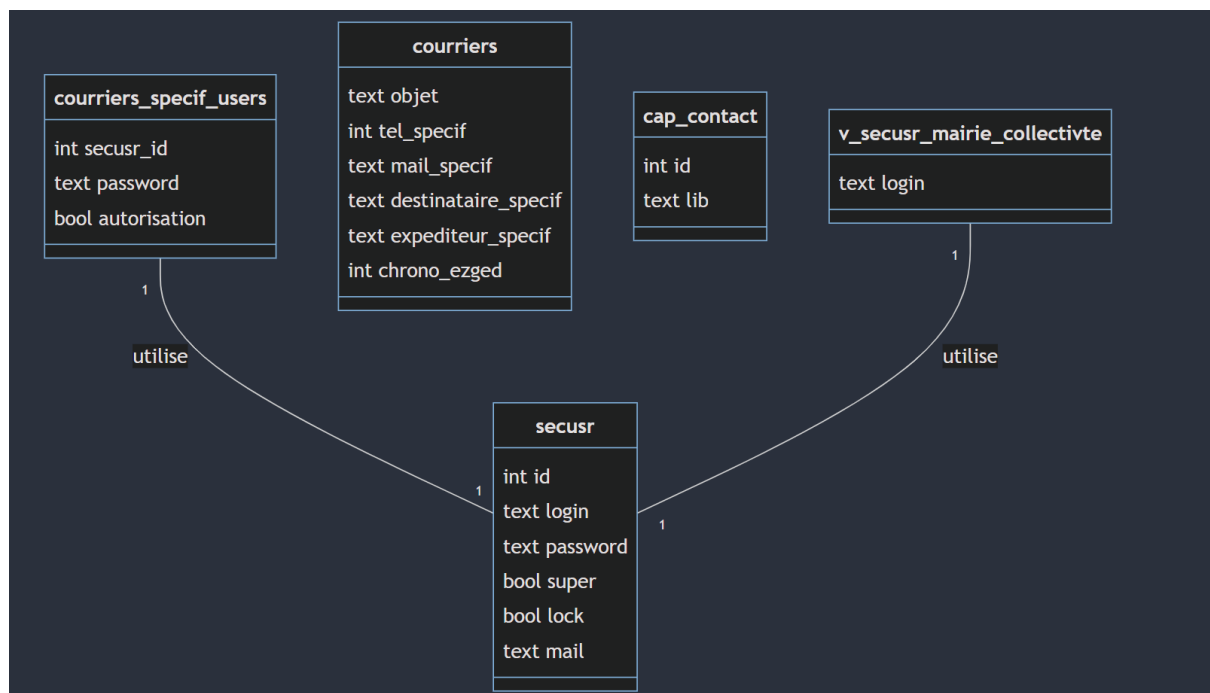
Cette table contient tous les expéditeurs qui ont l'habitude d'envoyer des courriers.

v_secusr_mairie_collectivte

login (texte)

Cette table permet de récupérer tous les utilisateurs de la table _secusr pour pouvoir les afficher dans le champ DESTINATAIRE

Pour résumer :



Il reste une table un peu à part car elle n'est pas vraiment propre à la GED c'est la table :

modele_courriers_specif

id (int)

lib (texte)

content (fulltxt)

Elle contient tous les modèles qui sont saisies depuis la page **creation_modele.php**

Création de modèle de courrier

Nom du modèle :

Contenu du modèle :

Créer le modèle

On les retrouvent dans le select de **formulaire_contact.php**

Sélectionnez un modèle de courrier ▼

Sélectionnez un modèle de courrier

demande de location de salle

demande de fermeture de route

Reconnaissance des champs :

Comme expliqué un peu plus tôt le formulaire de contact vient s'insérer dans le contenu finale du pdf pour obtenir un résultat semblable à celui-ci :


Expediteur :	CAP
Destinataire :	mc.maire
Email :	test@mail.fr
N° Tel :	0707070707
Objet :	Test

La GED dès qu'elle reçoit un document en fait un OCR, c'est-à-dire qu'elle le lit de bout en bout pour en récupérer tout le texte et le stocker dans un fichier txt. Ce fichier va être utilisé pour gérer le remplissage automatique des champs.

Le fait d'écrire clairement Objet : Test nous facilite la tâche pour récupérer les infos. Je précise simplement à ma GED de venir récupérer les 100 prochains caractères qui viennent après "Objet :." sans compter les espaces.

Je récupère donc dans notre exemple Test et je peux lui dire que ce texte doit être saisi dans mon champ objet de ma table qui est décrite dans la partie suivante.

Voila un aperçu de ce paramétrage sur la GED :

Type d'extraction:	Macros après un texte ▼
Description:	Tel
Champs destination:	COURRIERS_TEL_SPECIF ▼
Requis:	<input type="checkbox"/>
<hr/>	
Après ce texte:	N° Tel :
Nombre de caractères à extraire:	100 ▲▼
<hr/>	
Post formatage:	<input type="text"/> 
Gestion des espaces:	Supprimer aux extrémités e ▼

Il ne reste plus qu'à faire pareil pour tous les champs.

Ici j'ai fait en sorte que la récupération des champs se fasse facilement. Mais ce système peut permettre à l'aide d'expression régulière de venir récupérer des champs plus compliqués sur des factures par exemple

Requêtes SQL

Explications de toutes les requêtes SQL utilisés pour le bon déroulement de mon application :

autorisation.php :

Récupération de la liste des utilisateurs pour pouvoir les afficher dans le select

```
SELECT secusr_id, secusr_login FROM _secusr
```

Insertion du mot de passe pour accéder à l'outil de création de pdf en fonction de l'utilisateur sélectionné.

```
INSERT INTO courriers_specif_users (COURRIERS_SPECIF_USERS_SECUSR_ID, COURRIERS_SPECIF_USERS_PASSWORD) VALUES (:userId, :password)
```

Ajouter un mot de passe utilisateur

Utilisateur : Mot de passe :

formulaire_contact.php :

Récupérer les utilisateurs

```
SELECT CAP_CONTACT_LIB FROM cap_contact
```

Récupérer les destinataires

```
SELECT V_SECUSR_MAIRIE_COLLECTIVITE_LOGIN FROM v_secusr_mairie_collectivite
```

Sélectionnez un expéditeur

Sélectionnez un destinataire

Sélectionnez un destinataire

mc.maire

mc.dgs

mc.compta

mc.accueil

mc.agent

login.php :

Vérification des informations de connexions saisies par l'utilisateur

```
SELECT * FROM _secusr
```

```
INNER JOIN courriers_specif_users ON _secusr.SECUSR_ID =  
courriers_specif_users.COURRIERS_SPECIF_USERS_SECUSR_ID  
WHERE _secusr.SECUSR_LOGIN = :username AND  
courriers_specif_users.COURRIERS_SPECIF_USERS_PASSWORD = :password
```

formulaire_contact.php :

Récupération de tous les modèles de la table pour pouvoir ensuite les afficher et insérer dans la zone de saisie

```
SELECT * FROM modele_courriers_specif
```

creation_modele.php :

Insertion des infos saisies lors de la création d'un modèle

```
INSERT INTO modele_courriers_specif (MODELE_COURRIERS_SPECIF_LIB,  
MODELE_COURRIERS_SPECIF_CONTENT) VALUES (?, ?)
```

Bibliothèques utilisées

Présentation de TCPDF et Bootstrap

TCPDF :

TCPDF est une bibliothèque PHP open source utilisée pour générer des fichiers PDF dynamiques. Voici quelques points clés à retenir :

TCPDF offre une large gamme de fonctionnalités pour créer des documents PDF personnalisés.

Il prend en charge des éléments tels que les en-têtes et pieds de page, les tableaux, les images, les polices personnalisées, les liens et les annotations.

TCPDF permet de personnaliser les formats de page, l'orientation, la taille, la compression et le cryptage.

Bootstrap :

Bootstrap est un framework CSS populaire qui facilite la création d'interfaces utilisateur réactives et modernes. Voici quelques points clés à retenir :

Bootstrap fournit une collection de composants et de classes prédéfinis pour construire des sites web.

Il offre des fonctionnalités telles que des grilles, des formulaires, des boutons, des menus déroulants, des modales, des onglets, etc.

La conception réactive de Bootstrap permet à votre site web de s'adapter automatiquement à différents appareils et tailles d'écran.

Il est hautement personnalisable, vous permettant d'ajuster facilement l'apparence et le comportement des composants.

Description fonctions JS

Le fichier script.js contient l'ensemble de mes fonctions qui sont appelés pour pouvoir manipuler mon texte qui fera l'objet d'un PDF par la suite. Voici une explication pour chacune des fonctions :

[getSelectionParentElement\(\)](#) : Cette fonction permet de récupérer l'**élément parent*** de la sélection de texte. Elle utilise les objets JavaScript `window.getSelection()` ou `document.selection` pour récupérer la sélection de texte.

Elément parent* :

```
<div id="parent">  
  <p>Contenu du paragraphe</p>  
</div>
```

La div "parent" est parent du paragraphe `<p>` qui est donc enfant de la div "parent"

[surroundSelection\(tagName, styleAttribute\)](#) : Cette fonction permet d'ajouter des balises HTML autour de la sélection de texte. Elle prend en paramètres le nom de la balise et l'attribut de style (facultatif) à ajouter.

[bold\(\)](#) : Met en **gras** le texte sélectionné par [surroundSelection\(\)](#).

[underline\(\)](#) : Souligne le texte sélectionné par [surroundSelection\(\)](#).

[italic\(\)](#) : Met en *italique* le texte sélectionné par [surroundSelection\(\)](#).

[changeFontSize\(\)](#) : Change la taille du texte sélectionné par [surroundSelection\(\)](#). Le changement se fait en fonction de la taille choisie dans le menu déroulant.

[align\(type\)](#) : Aligne le texte sélectionné par [surroundSelection\(\)](#) à gauche ← ou au centre ou à droite →

Elle prend en paramètre le type d'alignement ("left" ou "center").

[prepareTextForPdf\(\)](#) : Cette fonction prépare le texte pour l'exportation au format PDF en modifiant l'espace entre chaque ligne pour que le rendu PDF soit mieux en ajoutant un attribut de style `line-height` aux paragraphes.

`changeTextColor()` : Change la **couleur** du texte sélectionné par `surroundSelection()`. Le changement se fait en fonction de la couleur choisie dans le menu.

`verifierExpéditeur()` : Vérifie si un nouvel expéditeur a été sélectionné dans le menu déroulant, et affiche ou masque un champ de saisie en conséquence. Champ dans lequel l'utilisateur devra saisir son nouvel expéditeur.

`ajouterInformations()` : Cette fonction crée un tableau HTML à partir des valeurs saisies dans les champs du formulaire, puis l'ajoute à la zone éditable.

Exemple :

Expéditeur : CHEVILLOTTE

Destinataire : mc.dgs

Email : test@mail.fr

N° Tél : 07864843064

Objet : test

`insertImage()` : Insère l'image sélectionnée par l'utilisateur dans la zone éditable en utilisant la méthode `range.insertNode()` qui permet d'insérer un élément à un emplacement donné, ici le curseur

`previewButton.onclick` : Ouvre une nouvelle fenêtre pour afficher la prévisualisation du fichier PDF.

Le script utilise également la bibliothèque jQuery pour manipuler le contenu HTML de la page avec une syntaxe plus concise que du javascript pur.

Exemple :

Dans la fonction `ajouterInformations()`, il est utilisé pour récupérer les valeurs des champs de formulaire et les ajouter dynamiquement avec les éléments html qui conviennent dans ma zone de texte. La méthode `$()` est utilisée pour sélectionner des éléments HTML, et les méthodes `append()` et `html()` sont utilisées pour ajouter du contenu HTML à la page.

Il y a également une partie de JS qui ne se trouve pas dans script.js mais dans `formulaire_contact.php`

Cette partie concerne la gestion des modèles avec la fonction `chargerModele()` elle permet de récupérer le modèle sélectionné pour l'ajouter à la zone de saisie de texte.

Les modèles sont contenus dans la variable **modeles**, ils font références à ceux de la table `modele_courriers_specif`

Description Gestion PDF

Le fichier gestion pdf.php est celui qui utilise la bibliothèque TCPDF et donc qui gère le PDF final, voici une explication de comment il fonctionne :

La fonction `isset($_POST['texte'])` vérifie si un texte a été soumis via le formulaire. Si c'est le cas, le texte est récupéré et stocké dans la variable \$texte.

La boucle `for` suivante ajoute un certain nombre de sauts de ligne (`
`) au début du texte. Cela permet de garantir un espace suffisant pour l'entête du document PDF.

Ensuite, on instancie un nouvel objet PDF avec `$pdf = new TCPDF()`. On ajoute une page au document PDF avec `$pdf->AddPage()` et on définit la police utilisée dans le document avec `$pdf->SetFont('times', '', 12)`.

La méthode `SetAutoPageBreak(false, 0)` désactive les sauts de page automatiques. C'est important car nous allons ajouter une image de fond au document et nous ne voulons pas qu'elle soit interrompue par un saut de page.

La méthode `$pdf->Image()` ajoute une image de fond au document PDF. Les paramètres spécifient l'emplacement de l'image, ses dimensions, et d'autres options.

La méthode `$pdf->writeHTML()` écrit le texte (avec les sauts de ligne ajoutés précédemment) dans le document PDF.

Ensuite, le code vérifie si une certaine option ("previsu") a été sélectionnée dans le formulaire POST avec `isset($_POST['previsu'])`. En fonction du résultat, le PDF est enregistré dans un dossier spécifique.

La méthode `$pdf->Output()` enregistre le fichier PDF sur le serveur. Le premier paramètre est le chemin complet du fichier, et le second paramètre 'F' signifie que le document doit être sauvegardé sur le disque.

Si l'option "previsu" a été sélectionnée, le code utilise ensuite une série de fonctions `header()` pour envoyer des en-têtes HTTP appropriés au navigateur de l'utilisateur. Cela permet au navigateur d'ouvrir le fichier PDF directement. La fonction `readfile()` lit le fichier PDF et l'envoie au navigateur.

Si l'option "previsu" n'a pas été sélectionnée, un message de confirmation est affiché à l'utilisateur avec le message de succès (car c'est qu'il a coché générer).

Version Mobile

L'entièreté de ma solution est responsive et fonctionne donc sur mobile, les outils pour styliser le texte sont par contre un peu moins simples d'utilisation mais les fonctionnalités principales restent simples d'utilisation.

C'est l'utilisation de bootstrap comme précisé avant qui permet une adaptation.

Il était important que la solution fonctionne sur mobile.

En effet, si le but est de générer un courrier suite à un entretien oral comme un appel téléphonique, il est facilement imaginable que l'utilisateur se connecte à l'outil directement après depuis son smartphone.

Dans le cas où il est en déplacement professionnel par exemple.

Administrateur

Pour terminer cette documentation je tenais à rajouter une partie sur le rôles administrateurs dans l'outil.

Un utilisateur lambda se connecte et rédige son courrier pour l'envoyer dans la GED. Mais l'administrateur a 2 rôles supplémentaires.

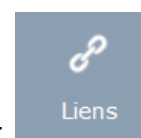
Il est chargé de dire qui est autorisé à utiliser l'outil en leur créant un mot de passe via **autorisation.php** que l'on a vu plus haut.

Je précise qu'il peut revenir sur son choix en décochant la case depuis la GED

NOM	AUTORISATION (
<input type="text"/>	<input type="text"/>
Poncet Celio	<input checked="" type="checkbox"/>


Mais il a également la charge de devoir saisir les modèles pour les courriers via **creation_modele.php** qui a aussi été vu avant.


Pour ne pas avoir à utiliser plusieurs lien différent il un onglet





sur la ged qui regroupe toutes les pages qu'il peut utiliser


▼ Mes liens


 Action rapide

 Autorisation Courriers

 Collectivité - Action rapide

 Création de courriers

 Demo des spécifiques

 Modeles courriers