1. MODALITÉS

→ Pondération : **25%** de la note finale du cours
→ Échéance de remise : Vendredi **25 mars** 2022 à **23h50**
→ Travail à réaliser en **équipe de 2**. Attention, le professeur peut, s'il le juge nécessaire, poser des questions à chacun des coéquipiers en regard au code informatique remis afin de s'assurer qu'ils sont les auteurs authentiques du travail remis et qu'ils ont tous les deux participé à la totalité de la conception et de la programmation. Ainsi, Il vous faudra, à tous les deux, maitriser l'ensemble du code qui aura été remis et avoir collaboré à la production de ce dernier.
→ Tout **plagiat** sera **sanctionné** selon les modalités prévues à la politique en vigueur à **Polytechnique Montréal**.

2. ÉNONCÉ

Pour ce travail pratique, concevoir une version évoluée et fonctionnelle de l'application web dynamique développée dans le cadre du travail pratique précédent.

Les modules / pages écrans ci-dessous, au terme de ce travail pratique, deviendront fonctionnels :

1- Rapport d'interventions
2- Rapport de déclaration d'une intervention
3- Mise-à-jour d'un rapport d'intervention
4- Retrait d'un rapport d'intervention

Pour ce faire, il vous faudra implémenter les fonctionnalités énumérées dans les différentes parties de l'énoncé.

Une base au présent travail pratique est gracieusement fournie en annexe. Cette base est le résultat de l'exécution initiale par Flask du code Python du solutionnaire préalablement préparé par le professeur. Ainsi, vous serez en mesure d'y voir toute la structure HTML générée lors de l'exécution initiale ainsi que le code JavaScript permettant de gérer l'interactivité de la page.

Noter que le module / page-écran « Visualisation » demeure non-fonctionnel (retranché du travail pour cause d'allégement).
Noter qu'il est **strictement interdit** de modifier les scripts « nav.js », « rapport.js » et « forms.js ». Il faudra les exploiter tels quels.

2.1. Partie 1 : page-écran « Le rapport d'interventions »

L'objet littéral JavaScript représentant le nombre d'interventions par poste de quartier que vous aviez conçu précédemment, devait être mis à jour fréquemment afin de fournir un portrait actuel et représentatif des interventions policières. Pour vous, ce travail manuel chronophage n'est plus une option, surtout que maintenant, vous avez appris à lire le contenu de documents structurés en Python (avec ou sans pandas). Vous décidez donc de mettre à profit cet apprentissage en agrémentant votre application web de code Python qui lira automatiquement les fichiers sources et générera le rapport HTML dynamiquement à chaque exécution, et ce, sans aucune intervention humaine (en utilisant le moteur de template pour Python « Jinja »).
Les fichiers ci-dessous contenant les données se trouvent dans le sous répertoire « data »:

- **Base/data/interventions.tsv (Extrait)**

interventions.tsv

```
ID_INTERVENTION DATE_INCIDENT   CATÉGORIE   PDQ QUART_TRAVAIL
1   2018-09-13  Vol de véhicule à moteur    30  jour
2   2018-04-30  Vol de véhicule à moteur    30  jour
3   2018-09-01  Vol de véhicule à moteur    7   nuit
4   2017-07-21  Méfait  21  jour
5   2017-07-29  Méfait  12  jour
6   2017-07-30  Vol de véhicule à moteur    21  nuit
7   2017-07-30  Méfait  38  jour
8   2017-07-30  Vols qualifiés  21  jour
9   2017-08-01  Vol dans / sur véhicule à moteur    39  jour
10  2017-08-01  Méfait  21  jour
11  2017-08-01  Vol de véhicule à moteur    5   jour
12  2018-08-28  Introduction    7   soir
13  2018-01-10  Introduction    30  jour
14  2018-11-12  Méfait  30  soir
15  2018-08-15  Méfait  30  jour
```

- **Base/data/pdq.csv (Extrait)**

pdq.csv

```
PDQ;EMPLACEMENT
1;Baie-D'Urfé, Beaconsfield, Kirkland, Sainte-Anne-de-Bellevue, Senneville
3;L'Île-Bizard, Pierrefonds, Sainte-Geneviève, Roxboro
4;Dollard-Des Ormeaux
5;Dorval, L'Île-Dorval, Pointe-Claire
7;Saint-Laurent
8;Lachine, Saint-Pierre
9;Côte Saint-Luc, Hampstead, Montréal-Ouest
10;Bordeaux, Cartierville
11;Notre-Dame-de-Grâce
12;Ville-Marie Ouest, Westmount
13;LaSalle
15;Saint-Paul, Petite-Bourgogne, Pointe-Saint-Charles, Saint-Henri, Ville-Émard
```

2.1.1. Vous devez maintenant créer dynamiquement le **corps du tableau HTML** en parcourant dynamiquement les structures Python appropriées, que vous aurez préalablement créés et rempli de données provenant des fichiers de données.
Dans le fichier de base fourni «Base_TP3.html», les lignes 179 à 212 (inclusivement) ont été générées dynamiquement en Python / Jinja.

2.1.2. Aussi, afin de permettre au script « rapport.js » de réaliser le calcul de statistiques ainsi que de surligner les rangées du tableau concernées par ces statistiques, l'objet JavaScript « nbInterventionsParPDQ » devra être généré par votre script Python à partir des données lues dans les fichiers de données.
Dans le fichier de base fourni «Base_TP3.html», les lignes 74 à 109 (inclusivement) ont été générées dynamiquement en Python / Jinja.

2.1.3. Le contenu du pied de page, notamment les deux dates utilisées par le script JavaScript, devront également être générés dynamiquement par votre Script Python. Ces dates représentent les interventions la plus ancienne et la plus récente qui sont consignées dans le fichier « **interventions.tsv** ».

Remarquer les deux déclaration JS « dateMin » et « dateMax » dans le fichier de base fourni.
Dans le fichier de base fourni «Base_TP3.html», les lignes 111 à 112 (inclusivement) ont été générées dynamiquement en Python / Jinja.

2.2. Partie 2 : page-écran « Rapport de déclaration d'une intervention »

2.2.1. Remplissage dynamique des champs du formulaire :

Toutes les valeurs des éléments du formulaire doivent être générées dynamiquement à partir de votre script Python à partir des données lues dans les fichiers de données.

2.2.1.1. Les options de la liste déroulante « # Poste de quartier » doivent être générées dynamiquement.
Dans le fichier de base fourni « Base_TP3.html », les lignes 225 à 258 (inclusivement) ont été générées dynamiquement en Python / Jinja.

2.2.1.2. Les options de la liste déroulante « Catégorie » doivent être générées dynamiquement.
Dans le fichier de base fourni « Base_TP3.html », les lignes 270 à 275 (inclusivement) ont été générées dynamiquement en Python / Jinja.

2.2.1.3. Les options de la liste déroulante « Quart » doivent être générées dynamiquement.
Dans le fichier de base fourni « Base_TP3.html », les lignes 291 à 293 (inclusivement) ont été générées dynamiquement en Python / Jinja.
De plus votre code Python devra ajouter dynamiquement l'attribut « selected » au quart de travail correspondant à l'heure actuelle de chargement de l'interface. Cette sélection devra se baser sur l'heure actuelle du système et la comparer aux heures de début et de fin des quarts de travail puisées dynamiquement dans la source de données « **Base/data/quarts_travail.csv** ».

quarts_travail.csv
```
ID;LIBELLÉ;HEURE_DEBUT;HEURE_FIN
1;jour;08:01:00;16:00:59
2;soir;16:01:00;00:00:59
3;nuit;00:01:00;08:00:59
```

Indice : voir la documentation pour récupérer l'heure courante ainsi que pour faciliter la comparaison de temps.

2.2.1.4. Les champs « Emplacement » et « Description » sont automatiquement mis à jour par le script « forms.js » pour refléter les valeurs de « # Poste de quartier » et « Catégorie » respectivement pour chaque changement.
Cependant les données utilisées par le script « forms.js » doivent être générés par votre code Python. À savoir, les objets « emplacementsPDQ » et « catInterventions ».

Dans le fichier de base fourni « Base_TP3.html », les lignes 16 à 51 (inclusivement) ont été générées dynamiquement en Python / Jinja.
Dans le fichier de base fourni « Base_TP3.html », les lignes 53 à 66 (inclusivement) ont été générées dynamiquement en Python / Jinja.

Note : La date affichée par défaut est celle du jour courant et est prise en charge par le script « forms.js ». Vous n'avez donc pas à implémenter cet affichage dynamique de valeur.

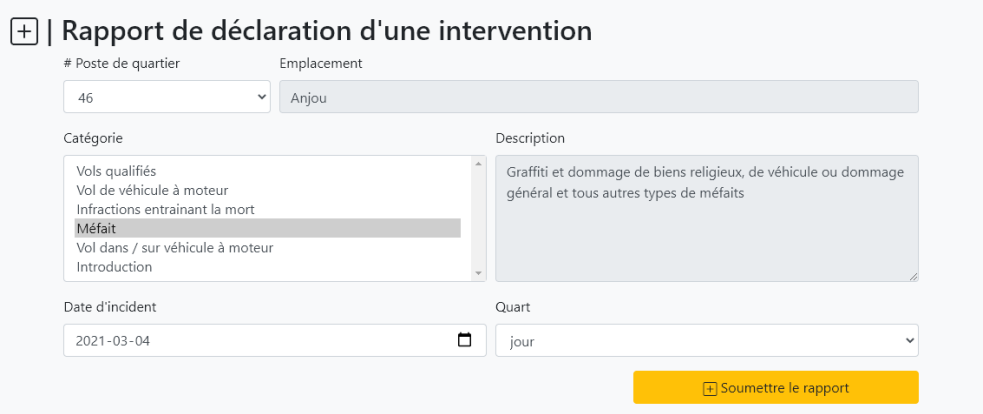
2.2.2. Soumission du rapport – enregistrement de la déclaration :

2.2.2.1. Au clic sur le bouton de soumission, l'enregistrement de la nouvelle déclaration est réputé être demandé et les données sont soumises via l'URL. Au rechargement de la page, lorsqu'une demande d'enregistrement de déclaration a été initiée, les données transmises sont récupérées et mises bout à bout afin de former une ligne de texte (selon le format des lignes de « **interventions.tsv** »). S'en suivra un enregistrement de la ligne à la fin du fichier.

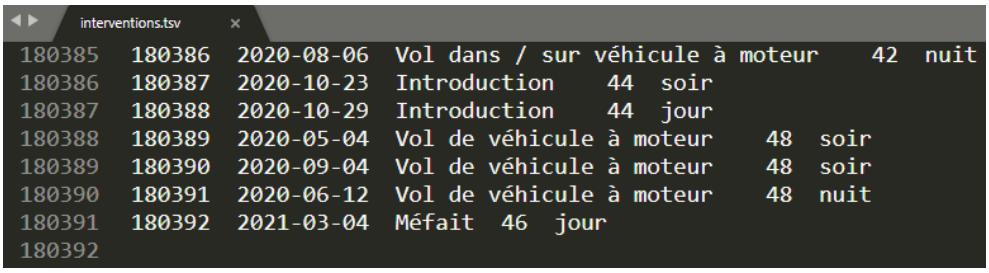
Extrait de fin de contenu de la source « interventions.csv » avant l'enregistrement d'une nouvelle déclaration :

interventions.tsv
```
180385  180386  2020-08-06  Vol dans / sur véhicule à moteur    42  nuit
180386  180387  2020-10-23  Introduction    44  soir
180387  180388  2020-10-29  Introduction    44  jour
180388  180389  2020-05-04  Vol de véhicule à moteur    48  soir
180389  180390  2020-09-04  Vol de véhicule à moteur    48  soir
180390  180391  2020-06-12  Vol de véhicule à moteur    48  nuit
180391
```

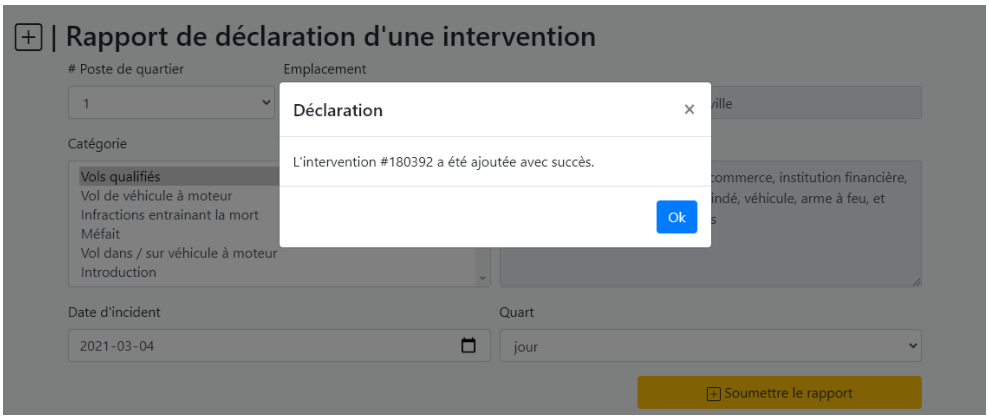
Demande d'enregistrement d'une nouvelle déclaration :



Extrait de fin de contenu de la source « interventions.csv » après la demande l'enregistrement de la nouvelle déclaration :



2.2.2.2. Suite à l'enregistrement fructueux, une fenêtre modale confirmant cet enregistrement est affichée et le formulaire de déclaration réaffiché et réinitialisé avec les valeurs par défaut.



[1]

Pour ce faire, le script JS suivant devra être dynamiquement injecté par le code Python/Jinja au moment opportun. Attention le numéro « #180392 » aura été généré automatiquement par votre script et est le résultat du dernier numéro d'intervention consignée incrémenté de 1.

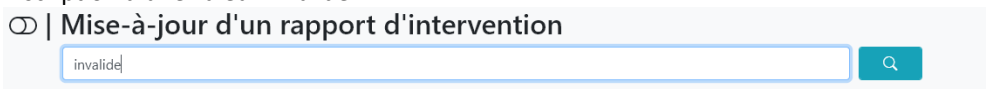
```
<script>
//Une déclaration a été ajoutée, afficher le popup de confirmation
window.addEventListener("load", ()=> {
        document.getElementById("popup-title").textContent   = "Déclaration";
        document.getElementById("popup-message").textContent = "L'intervention #180392 a été ajoutée avec succès.";

        //JQuery pour l'affichage de la fenêtre modale Bootstrap
        $("#popup").modal();

        //Lorsque la fenêtre modale est fermée, recharger la page web en s'assurant d'enlever tous les paramètres dans la
        //requête
        $("#popup").on('hidden.bs.modal', (e) => {
                window.location.search = "";
        })
});
</script>
```

2.3. Partie 3 : page-écran « Mise-à-jour d'un rapport d'intervention »

2.3.1. Formulaire de recherche de rapport préexistant :

2.3.1.1. La soumission d'une valeur invalide provoquera le réaffichage du même formulaire de recherche adjoint d'une fenêtre modale Bootstrap tels que présentés ci-dessous.

Inscription d'une valeur invalide :

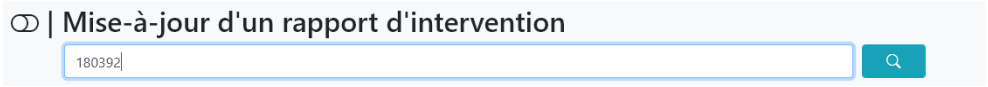


Affichage conséquent :

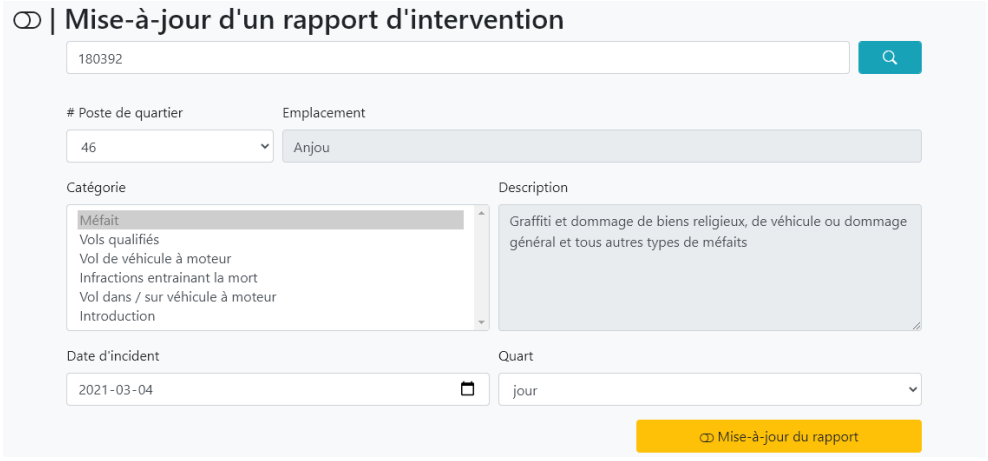
Pour ce faire, le script JS suivant devra être dynamiquement injecté par le code Python/Jinja au moment opportun. Dans l'exemple ci-dessus, la valeur « invalide » affichée dans la fenêtre modale, reflète le numéro erroné inscrit par l'utilisateur.

```
<script>
//Une déclaration a été ajoutée, afficher le popup de confirmation
window.addEventListener("load", ()=> {
        document.getElementById("popup-title").textContent   = "Erreur";
        document.getElementById("popup-message").textContent = "L'intervention #invalide n'existe pas.";

        //JQuery pour l'affichage de la fenêtre modale Bootstrap
        $("#popup").modal();

        //Lorsque la fenêtre modale est fermée, recharger la page web en s'assurant d'enlever tous les
        //paramètres dans la requête.
        $("#popup").on('hidden.bs.modal', (e) => {
                window.location.search = "";
        })
});
</script>
```

2.3.1.2. La soumission d'une valeur existante « valide » provoquera l'affichage du formulaire de modification pré rempli de valeurs correspondantes à celles de l'intervention chargée (voir la prochaine section : 2.3.2).



Affichage conséquent :



En effet, voici l'enregistrement correspondant au #180392 dans la source « interventions.csv » :

```
180392  2021-03-04  Méfait  46  jour
```

2.3.1.3. Suite à une requête de recherche pour un rapport d'intervention, afin de statuer sur l'affichage du formulaire de modification, le script « forms.js » utilise une variable booléenne « validInterventionNbToModify » générée préalablement par votre script Python lui indiquant s'il s'agit bien d'un numéro d'intervention valide. La valeur « true » affectée à cette variable JavaScript provoquera, en temps opportun, l'affichage du formulaire de mise-à-jour.

2.3.2. Remplissage dynamique des champs du formulaire et présélection des valeurs correspondantes à aux informations enregistrées :

Toutes les valeurs des éléments du formulaire doivent être générées dynamiquement en Python à partir des données lues dans les fichiers de données.

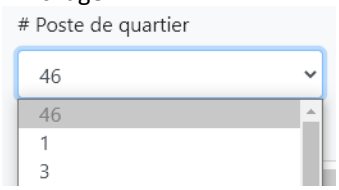
2.3.2.1. Le numéro de l'intervention à modifier doit demeurer affiché dans la zone de recherche
Dans le fichier de base fourni « Base_TP3.html », la déclaration de l'attribut « value » et sa valeur à la ligne 311 ont été générées dynamiquement en Python / Jinja.

```
<input  type        = "text"
        class       = "form-control"
        id          = "modify_no_intervention"
        name        = "modify_no_intervention"
        placeholder = "Inscrire le # du rapport d'intervention à mettre à jour"
        value       = "180392">
```

2.3.2.2. Les options de la liste déroulante « # Poste de quartier » doivent être générées dynamiquement.
Dans le fichier de base fourni « Base_TP3.html », les lignes 324 à 357 (inclusivement) ont été générées dynamiquement en Python / Jinja.
L'option représentant la valeur actuelle du numéro de poste de quartier doit être présélectionnée et apparaissant en premier lieu parmi l'ensemble des options. Cette option devra également ne pas être sélectionnable de nouveau.

```
<option value="46" selected disabled>46</option>
```

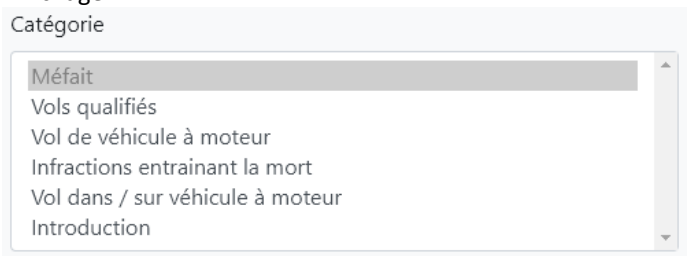
Affichage :



2.3.2.3. Les options de la liste déroulante « Catégorie » doivent être générées dynamiquement.
Dans le fichier de base fourni « Base_TP3.html », les lignes 369 à 374 (inclusivement) ont été générées dynamiquement en Python / Jinja.
L'option représentant la valeur actuelle de la catégorie d'intervention doit être présélectionnée et apparaissant en premier lieu parmi l'ensemble des options. Cette option devra également ne pas être sélectionnable de nouveau.

```
<option value="Méfait" selected disabled>Méfait</option>
```

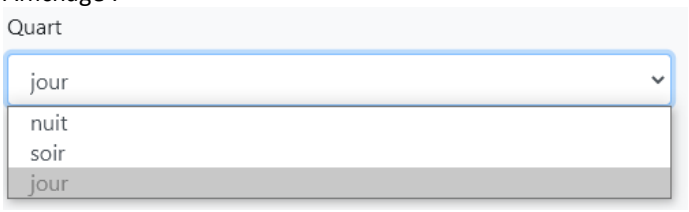
Affichage :



2.3.2.4. Les options de la liste déroulante « Quart » doivent être générées dynamiquement.
Dans le fichier de base fourni « Base_TP3.html », les lignes 390 à 392 (inclusivement) ont été générées dynamiquement en Python / Jinja.
L'option représentant la valeur actuelle du quart de travail doit être présélectionnée. Cette option devra également ne pas être sélectionnable de nouveau.

```
<option value="3" >nuit</option>
<option value="2" >soir</option>
<option value="1" selected disabled>jour</option>
```

Affichage :

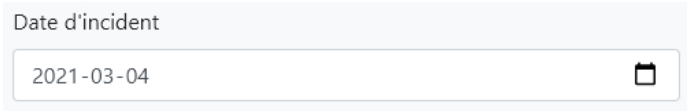


2.3.2.5. Même que 2.2.1.4

2.3.2.6. La date affichée correspond à la valeur lue dans la source de donnée. Elle est donc générée dynamiquement en Python / Jinja.

```
<input  type = "date"
        class = "form-control"
        id    = "modify_date_incident"
        name  = "modify_date_incident"
        value = '2021-03-04'>
```

Affichage :



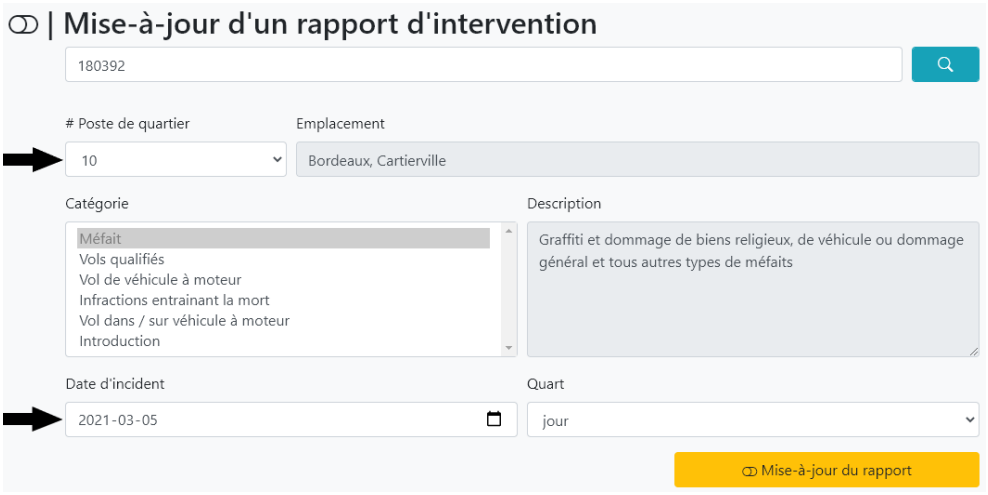
2.3.3. Soumission du rapport – mise-à-jour de la déclaration :

2.3.3.1. Au clic sur le bouton de soumission, l'enregistrement de la nouvelle version de déclaration (modifiée) est réputé être demandé et les données sont soumises via l'URL.
Un code JavaScript interrompant cette soumission afin de renseigner l'ensemble des valeurs (même celles demeurant inchangées, a été gracieusement fourni dans le script « forms.js », voir la méthode « modifyIntervention »).
À la toute fin, cette même méthode JavaScript provoquera un rechargement de la page en ayant préalablement renseigné l'ensemble des valeurs à même les paramètres de l'URL. Pour ce faire, elle exploite le contenu de l'objet JavaScript « interventionToModify » généré dynamiquement dans le script Python / Jinja.

```
<script>
interventionToModify = {
    "4" : "jour"   ,
    "3" : "10"     ,
    "2" : "Méfait" ,
    "1" : "2021-03-05"
};
</script>
```

Vous devez donc vous assurer de générer cet objet JavaScript dynamiquement dans votre script Python / Jinja afin que ce dernier soit prêt à être exploité au moment opportun par le script JavaScript. Les valeurs des différentes propriétés reflètent bien entendu les valeurs qui caractérisent l'intervention mise-à-jour.

Voici les changements apportés par l'agent de police sur l'interface qui ont provoqués les valeurs dans l'objet JavaScript ci-dessus :



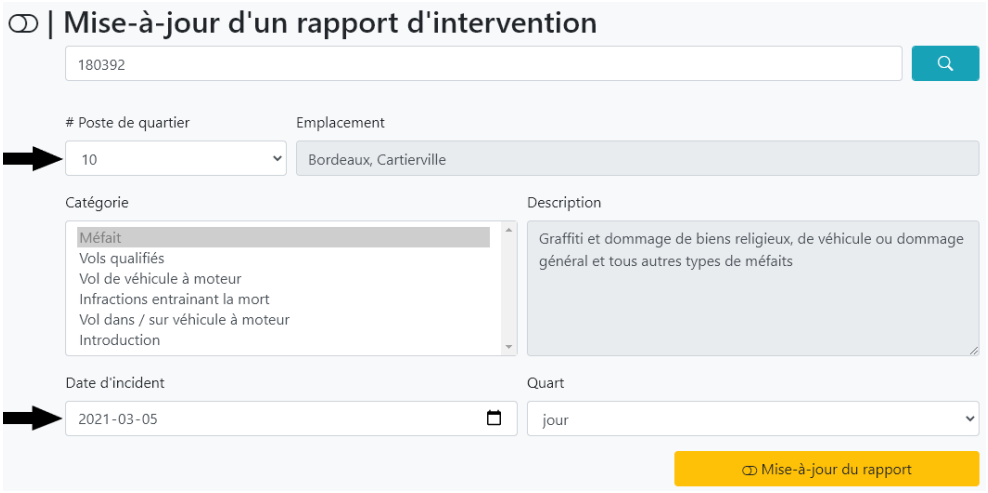
Le rechargement de la page, provoqué par la méthode « modifyIntervention » déclenchera les actions ci-dessous.

2.3.3.2. L'enregistrement de la ligne représentant les nouvelles valeurs de l'intervention dans le fichier.

Ligne de donnée dans « interventions.csv » avant la demande de modification :

```
180392  2021-03-04  Méfait  46  jour
```

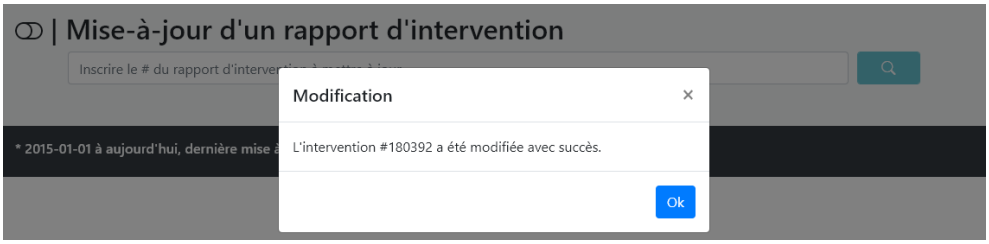
Demande d'enregistrement d'une nouvelle déclaration :



Ligne de donnée dans « interventions.csv » suite à la demande de modification :

```
180392  2021-03-05  Méfait  10  jour
```

2.3.3.3. Suite à l'enregistrement fructueux, une fenêtre modale confirmant cet enregistrement est affichée et le formulaire de recherche d'intervention à des fins de mise-à-jour réaffiché et réinitialisé.



Pour ce faire, le script JS suivant devra être dynamiquement injecté par le code Python au moment opportun.

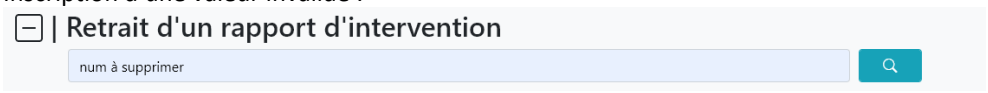
```
<script>
//Une déclaration a été ajoutée, afficher le popup de confirmation
window.addEventListener("load", ()=> {
        document.getElementById("popup-title").textContent   = "Modification";
        document.getElementById("popup-message").textContent = "L'intervention #180392 a été modifiée avec succès.";

        //JQuery pour l'affichage de la fenêtre modale Bootstrap
        $("#popup").modal();

        //Lorsque la fenêtre modale est fermée, recharger la page web en s'assurant d'enlever tous les paramètres dans
        //la requête
        $("#popup").on('hidden.bs.modal', (e) => {
                window.location.search = "";
        })
});
</script>
```

2.4. Partie 4 : page-écran « Retrait d'un rapport d'intervention »

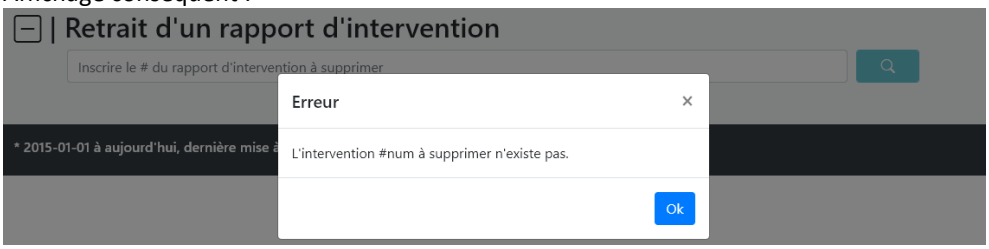
2.4.1. Formulaire de recherche de rapport préexistant :

2.4.1.1. La soumission d'une valeur invalide provoquera le réaffichage du même formulaire de recherche adjoint d'une fenêtre modale Bootstrap tels que présentés ci-dessous.

Inscription d'une valeur invalide :



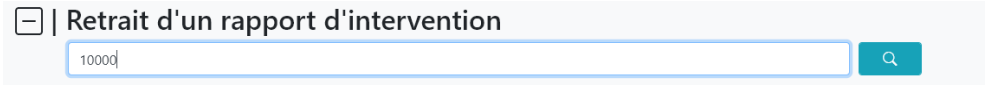
Affichage conséquent :



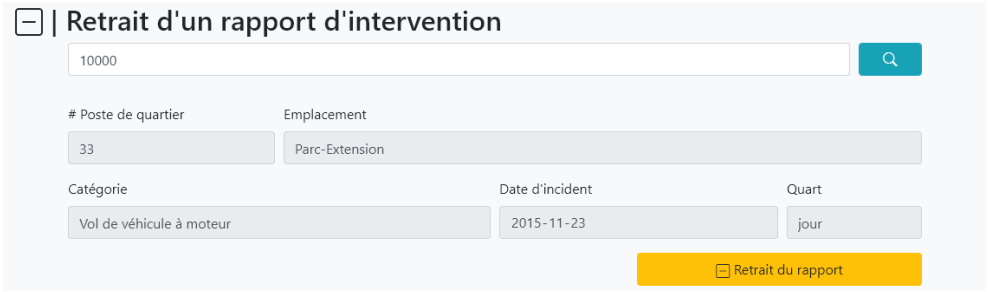
Pour ce faire, le script JS suivant devra être dynamiquement injecté par le code Python/Jinja au moment opportun. Dans l'exemple ci-dessus, la valeur « num à supprimer » affichée dans la fenêtre modale, reflète le numéro erroné inscrit par l'utilisateur.

```
<script>
//Une déclaration a été ajoutée, afficher le popup de confirmation
window.addEventListener("load", ()=> {
        document.getElementById("popup-title").textContent   = "Erreur";
        document.getElementById("popup-message").textContent =
                                                    "L'intervention #num à supprimer n'existe pas.";
        //JQuery pour l'affichage de la fenêtre modale Bootstrap
        $("#popup").modal();

        //Lorsque la fenêtre modale est fermée, recharger la page web en s'assurant d'enlever tous les
        //paramètres dans la requête.
        $("#popup").on('hidden.bs.modal', (e) => {
                window.location.search = "";
        })
});
</script>
```

2.4.1.2. La soumission d'une valeur existante « valide » provoquera l'affichage du formulaire de retrait pré rempli de valeurs correspondantes à celles de l'intervention chargée (voir la prochaine section : 2.4.2).



Affichage conséquent :



En effet, voici l'enregistrement correspondant au #10000 dans la source « interventions.csv » :

```
10000   2015-11-23  Vol de véhicule à moteur    33  jour
```

2.4.1.3. Suite à une requête de recherche pour un rapport d'intervention, afin de statuer sur l'affichage du formulaire de retrait, le script « forms.js » utilise une variable booléenne « validInterventionNbToRemove » générée préalablement par votre script Python lui indiquant s'il s'agit bien d'un numéro d'intervention valide. La valeur « true » affectée à cette variable JavaScript provoquera, en temps opportun, l'affichage du formulaire de retrait.

2.4.2. Remplissage dynamique des champs du formulaire avec les valeurs correspondantes à l'intervention telles que puisées dans la source de données :

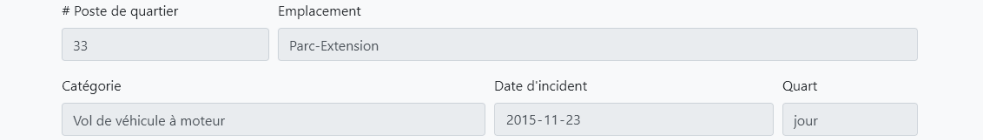
Toutes les valeurs des éléments du formulaire doivent être générées dynamiquement en Python/Jinja à partir des données lues dans les fichiers de données.

Tous les éléments de formulaire sont désactivés. En effet, cette page-écran sert à afficher, à titre indicatif, à l'utilisateur le rapport d'intervention qui est sur le point de supprimer de la source de données.

2.4.2.1. Le numéro de l'intervention à retirer doit demeurer affiché dans la zone de recherche
Dans le fichier de base fourni « Base_TP3.html », la déclaration de l'attribut « value » et sa valeur à la ligne 410 ont été générées dynamiquement en Python/Jinja.

```
<input  type        = "text"
        class       = "form-control"
        id          = "remove_no_intervention"
        name        = "remove_no_intervention"
        placeholder = "Inscrire le # du rapport d'intervention à supprimer"
        value       = "10000">
```

2.4.2.2. Les valeurs de chacun des champs de ce formulaire doivent être renseignées dynamiquement en Python/Jinja.

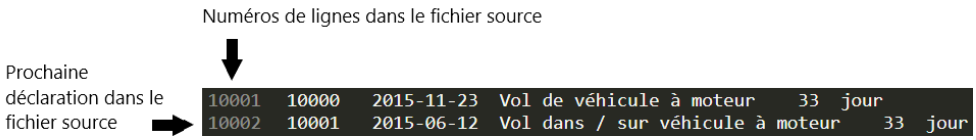


2.4.3. Retrait du rapport d'intervention:

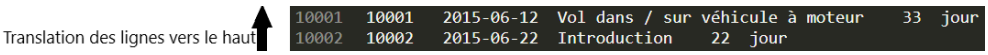
2.4.3.1. Au clic sur le bouton de soumission, le retrait du rapport d'intervention est demandé et son numéro est transmis via l'URL pour suppression effective.

Cette requête aura pour effet de provoquer la suppression de la ligne représentant l'intervention dans le fichier source.

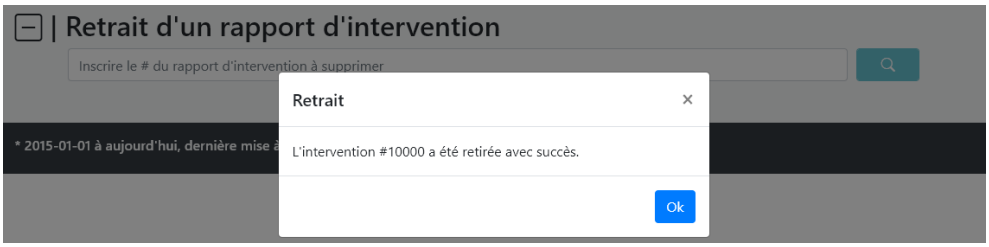
Lignes de données dans « interventions.csv » avant la requête de suppression :



Lignes de données dans « interventions.csv » après la requête de suppression :



2.4.3.2. Suite à la suppression fructueuse, une fenêtre modale confirmant ce retrait est affichée et le formulaire de recherche d'intervention à des fins de retrait réaffiché et réinitialisé.



Pour ce faire, le script JS suivant devra être dynamiquement injecté par le code Python/Jinja au moment opportun.

```
<script>
//Une déclaration a été ajoutée, afficher le popup de confirmation
window.addEventListener("load", ()=> {
        document.getElementById("popup-title").textContent   = "Retrait";
        document.getElementById("popup-message").textContent = "L'intervention #10000 a été retirée avec succès.";

        //JQuery pour l'affichage de la fenêtre modale Bootstrap
        $("#popup").modal();

        //Lorsque la fenêtre modale est fermée, recharger la page web en s'assurant d'enlever tous les paramètres dans
        //la requête
        $("#popup").on('hidden.bs.modal', (e) => {
                window.location.search = "";
        })
});
</script>
```

3. SPÉCIFICATIONS

Tel que demandé dans l'énoncé, l'utilisation du langage Python 3 et des cadriciels Pandas et Flask pour réaliser l'ensemble des éléments est requise. Aucune autre méthode utilisée ne sera rétribuée.
On vous demande de respecter le fonctionnement tel qu'énoncé dans le présent document.

4. CRITÈRES ET PROCÉDURE DE REMISE

Prenez soin de vous identifier en entête du document en inscrivant votre nom et prénom à deux endroits :

1. En commentaire HTML en haut du document selon le format ci-dessous :

```
<!--
        Étudiant 1 : Nom, Prénom (Matricule)
        Étudiant 2 : Nom, Prénom (Matricule)

        INF8007 – Langages de scripts
-->
```

2. Ainsi qu'à l'espace réservé en entête (Remplacez par vos noms et prénoms)

Une fois le travail complété, compresser l'ensemble des documents et remettre une seule copie de l'archive « **zip** » (**par l'un des coéquipiers**) sous l'intitulé **TP3 dans la section « Travaux pratiques à remettre » sur Moodle**. Attention aucune autre méthode de remise ne sera acceptée.

5. CRITÈRES DE CORRECTION

	Critères de correction	18 Points
P1 - Rapport	2.1.1. Génération du corps du tableau HTML	6
	2.1.2. Génération de l'objet JavaScript « nbInterventionsParPDQ »	8
	2.1.3. Génération du contenu de pied de page	4

	Critères de correction	25 Points
P2 - Déclaration	**2.1.1. Remplissage dynamique des champs du formulaire**	**Total /18**
	2.2.1.1. Les options de la liste déroulante « # Poste de quartier » générées dynamiquement.	3
	2.2.1.2. Les options de la liste déroulante « Catégorie » générées dynamiquement.	3
	2.2.1.3. Les options de la liste déroulante « Quart » générées dynamiquement + selected	6
	2.2.1.4. les objets JS « emplacementsPDQ » et « catInterventions » générés dynamiquement	6
	2.2.2. Soumission du rapport – enregistrement de la déclaration :	**Total / 7**
	2.2.2.1. Enregistrement de la nouvelle déclaration dans le fichier de données	5
	2.2.2.2. Fenêtre modale confirmant l'enregistrement est affichée	2

	Critères de correction	37 Points
P3 – Mise-à-jour	**2.3.1. Formulaire de recherche de rapport préexistant**	**Total / 6**
	2.3.1.1. Traitement de soumission d'une valeur invalide	2
	2.3.1.2. Traitement de soumission d'une valeur existante « valide »	2
	2.3.1.3. validInterventionNbToModify = true/false	2
	2.3.2. Remplissage dynamique des champs du formulaire et présélection des valeurs	**Total / 17**
	2.3.2.1. Numéro de l'intervention à modifier demeure affiché dans la zone de recherche	2
	2.3.2.2. Les options de la liste déroulante « # Poste de quartier » + 1ère + sélectionnée + désactivée	5
	2.3.2.3. Les options de la liste déroulante « Catégorie » + 1ère + sélectionnée + désactivée	5
	2.3.2.4. Les options de la liste déroulante « Quart » + sélectionnée + désactivée	4
	2.3.2.6. La bonne date affichée.	1
	2.3.3. Soumission du rapport – mise-à-jour de la déclaration	**Total / 14**
	2.3.3.1. Génération de l'objet JS « interventionToModify » avec les bonnes valeurs	5
	2.3.3.2. Mise-à-jour de la ligne représentant les nouvelles valeurs de l'intervention dans le fichier.	7
	2.3.3.3. Fenêtre modale confirmant la modification est affichée	2

	Critères de correction	20 Points
P4 – Retrait	**2.4.1. Formulaire de recherche de rapport préexistant**	**Total / 6**
	2.4.1.1. Traitement de soumission d'une valeur invalide	2
	2.4.1.2. Traitement de soumission d'une valeur existante « valide »	2
	2.4.1.3. validInterventionNbToRemove = true/false	2
	2.4.2. Remplissage dynamique des champs du formulaire et présélection des valeurs	**Total / 7**
	2.4.2.1. Numéro de l'intervention à retirer demeure affiché dans la zone de recherche	2
	2.4.2.2. Valeurs renseignées : PDQ + Emplacement + Catégorie + Date + Quart	5
	2.4.3. Retrait du rapport d'intervention	**Total / 7**
	2.4.3.1. Suppression de la ligne représentant le rapport d'intervention à retirer.	5
	2.4.3.2. Fenêtre modale confirmant la modification est affichée	2