

# PROJET DE MODÉLISATION

RAPPORT DE PROJET

CLUZEL OPHELIE & VAUBON SAMUEL

## RÉFÉRENDUM D'INITIATIVE PARTAGÉE

### INTRODUCTION



**Qu'est-ce que le référendum d'initiative partagée ?**

Le référendum d'initiative partagée est le dispositif prévu par l'article 11 de la Constitution depuis la révision constitutionnelle de 2008. Un référendum portant sur les domaines mentionnés à l'article 11 de la Constitution « peut être organisé à l'initiative d'un cinquième des membres du Parlement, soutenue par un dixième des électeurs inscrits sur les listes électorales ».

**Modalités de mise en œuvre**

Les modalités de mise en œuvre du référendum d'initiative partagée sont fixées par les articles 11 et 61 de la Constitution modifiés par la loi constitutionnelle du 23 juillet 2008, la loi organique n° 2013-1114 et la loi n° 2013-1116 du 6 décembre 2013 portant application de l'article 11 de la Constitution ainsi que le décret n° 2014-1488 du 11 décembre 2014 relatif au traitement automatisé de données à caractère personnel dénommé « Soutien d'une proposition de loi au titre du troisième alinéa de l'article 11 de la Constitution ».

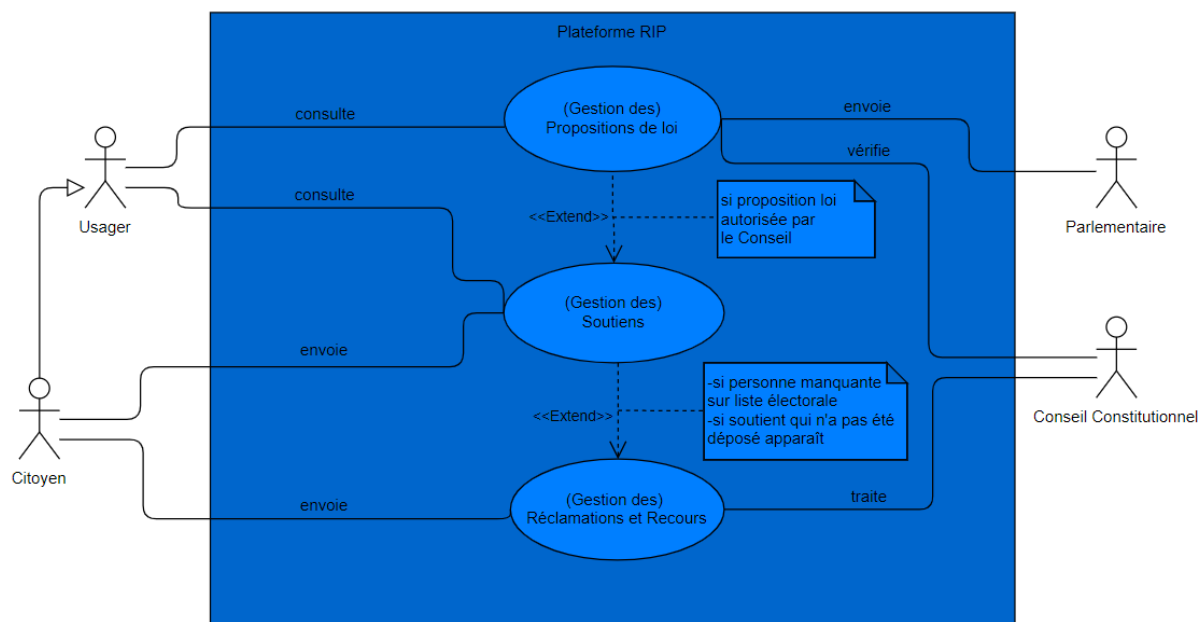
Le but de ce travail était de proposer différents documents (diagrammes UML, documents XML, représentation en base de données relationnelles), en utilisant les notions apprises en cours de modélisation et de base de données, permettant de décrire la plateforme RIP que l'on peut voir ci-dessus.

Cette plateforme de référendum d'initiative partagée a été mise en place par le gouvernement afin de permettre au corps électoral de soutenir différentes propositions de loi. Une proposition de loi est ouverte aux soutiens en ce moment concernant les Aéroports de Paris (ADP) et visant à affirmer le caractère de service public national de l'exploitation de ceux-ci.

## 1. PARTIE UML ET MODÈLE RELATIONNEL

### Description des fonctionnalités

#### DIAGRAMME USE CASE DE PREMIER NIVEAU REPRÉSENTANT LA PLATEFORME RIP



#### EXPLICATIONS

##### Acteurs :

Nous sommes partis du principe que n'importe qui pouvait accéder au site internet de la plateforme RIP on a donc créé un acteur « **Usager** » représentant ces personnes-là. Cependant certaines fonctionnalités ne pourront pas être effectuées par n'importe quel usager, notamment pour déposer un soutien il faut posséder le droit de vote et être inscrit sur les listes électorales : pour cette raison nous créons un acteur « **Citoyen** ». Ces acteurs sont les acteurs principaux car les fonctionnalités de la plateforme RIP vont leur rendre des services, pour cette raison on les représente à gauche du système (nous aurions également pu indiquer cela en ajoutant une balise « **primary** » en dessous de ces acteurs).

Il nous reste donc à voir les acteurs secondaires. Tout d'abord nous avons vu dans le sujet que ce sont les parlementaires qui peuvent proposer une loi, nous rajoutons donc un acteur « **Parlementaire** ». Il est dit également que les parlementaires sont composés notamment des républicains, des insoumis, des socialistes et des communistes, nous aurions pu rajouter des acteurs pour représenter cela mais dans notre cas ce n'était pas très utile et ça n'aurait fait que surcharger le

diagramme. Il est marqué sur le site que c'est le Conseil constitutionnel qui va vérifier les propositions de loi déposées par les parlementaires. C'est également lui qui va traiter les réclamations et les recours envoyés par les citoyens. On crée donc un dernier acteur « [Conseil Constitutionnel](#) ».

Cas d'utilisation :

Le site nous offre différentes fonctionnalités que nous avons regroupées dans 3 grands cas d'utilisation. Le premier cas d'utilisation est la « [gestion des propositions de loi](#) » qui regroupe les fonctionnalités concernant les propositions de loi : consulter ou proposer une nouvelle loi.

Le deuxième cas d'utilisation est la « [gestion des soutiens](#) » qui regroupe les fonctionnalités liées aux soutiens : déposer un soutien, consulter les soutiens, rechercher un soutien.

Pour finir le dernier cas d'utilisation est la « [gestion des réclamations et recours](#) » qui regroupe les fonctionnalités d'envoi et de traitement des réclamations et des recours.

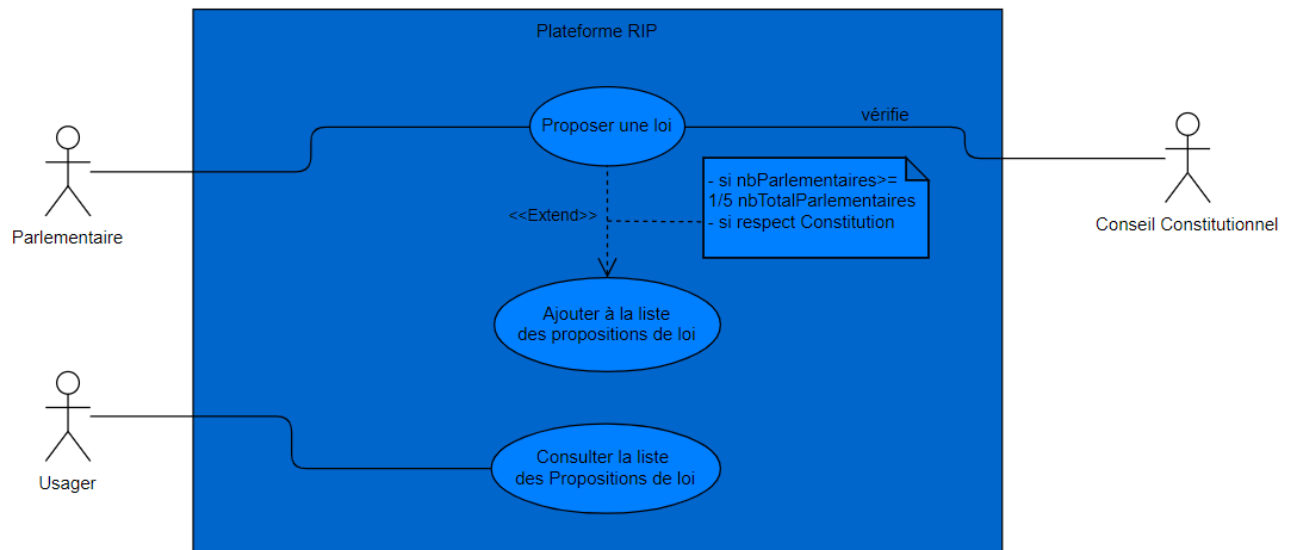
Relations :

Tout d'abord nous avons créé une relation de [généralisation](#) entre les acteurs « [Citoyen](#) » et « [Usager](#) » car même si le citoyen aura accès à certaines fonctionnalités que l'utilisateur n'aura pas, il pourra avoir accès également aux fonctionnalités destinées à n'importe quel utilisateur lambda. Nous aurions pu également rajouter d'autres relations de généralisation avec l'acteur utilisateur car dans l'absolu un parlementaire ou le Conseil Constitutionnel peuvent aussi être utilisateurs du site mais ça n'avait pas beaucoup de sens de le rajouter dans notre cas. Ainsi un utilisateur (et donc un citoyen également) pourra consulter la liste des propositions de loi et le recueil de soutiens, mais seul un citoyen pourra déposer un soutien et envoyer une réclamation ou un recours. Un parlementaire quant à lui pourra envoyer une proposition de loi et le Conseil Constitutionnel pourra vérifier les propositions de loi et traiter les réclamations et recours.

Enfin nous avons ajouté une relation d'[extension](#) entre les cas d'utilisation gestion des propositions de loi et gestion des soutiens car si jamais aucune proposition de loi n'est acceptée par le Conseil, alors il n'y aura pas de recueil de soutiens associé. Nous n'avons pas mis le point d'extension mais logiquement l'extension se ferait après qu'une loi ait été proposée. Nous avons ajouté une seconde relation d'[extension](#) entre les cas d'utilisation gestion des soutiens et gestion des réclamations et recours car premièrement s'il n'y a pas de recueil de soutiens il n'y aura pas de réclamations à faire et deuxièmement s'il n'y a pas de problèmes liés aux soutiens (personne manquante ou en trop dans la liste par exemple) il n'y aura pas besoin de faire de réclamations non plus.

---

## DIAGRAMME USE CASE DE 2<sup>EME</sup> NIVEAU REPRÉSENTANT LA FONCTIONNALITÉ GESTION DES PROPOSITIONS DE LOI



---

### EXPLICATIONS

#### Acteurs :

Nous retrouvons les mêmes acteurs que dans le diagramme de premier niveau. L'acteur « **Parlementaire** » devient ici un acteur principal.

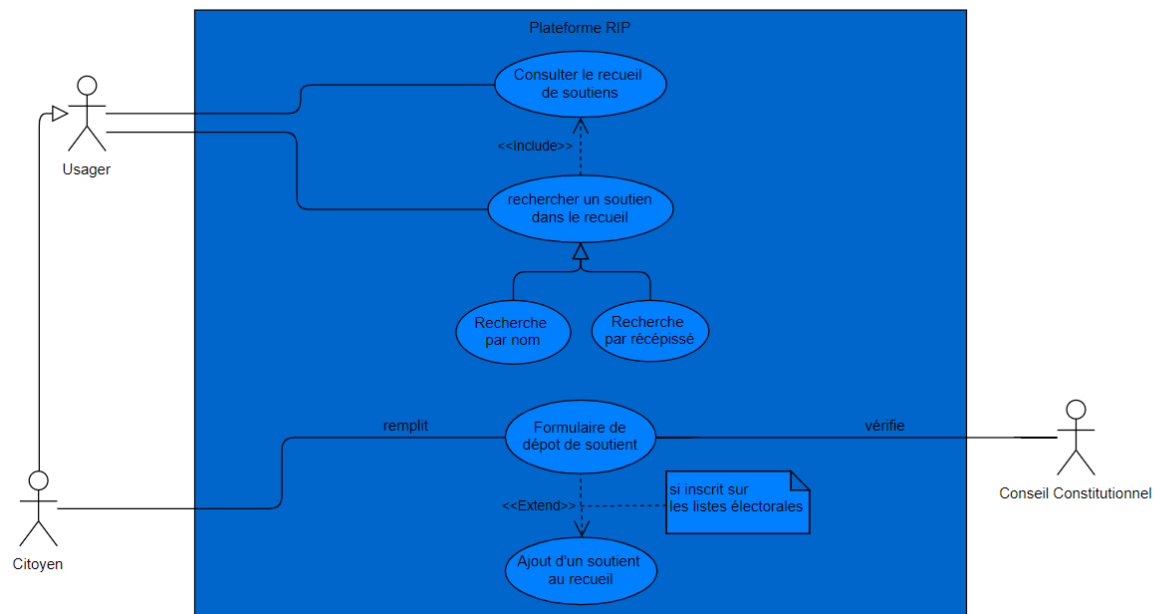
#### Cas d'utilisation :

La fonctionnalité « **Gestion des propositions de loi** » est composée de 3 cas d'utilisation. Tout d'abord **proposer une loi**, nous sommes partis du principe que les parlementaires devaient passer par la plateforme pour faire leurs propositions, dans la réalité ce n'est sûrement pas le cas. Ensuite on peut **ajouter une proposition de loi** à la liste des propositions de loi et enfin on peut **consulter** cette liste.

#### Relations :

Ce sont les parlementaires qui pourront proposer une loi, mais cette loi ne sera ajoutée à la liste des propositions de loi que si le nombre de parlementaires ayant proposé cette loi est supérieur à 1/5<sup>eme</sup> du nombre total de parlementaires et si le Conseil Constitutionnel juge que cette proposition respecte bien la Constitution. Nous avons donc mis une relation d'**extension** entre les cas d'utilisation proposer et ajouter une loi car si la proposition n'est pas acceptée, la loi ne sera pas ajoutée. Enfin pour finir n'importe quel usager pourra consulter la liste des propositions de loi.

## DIAGRAMME USE CASE DE 2<sup>EME</sup> NIVEAU REPRÉSENTANT LA FONCTIONNALITÉ GESTION DES SOUTIENS



### EXPLICATIONS

#### Acteurs :

Nous retrouvons ici encore les mêmes acteurs que dans le diagramme de premier niveau.

#### Cas d'utilisation :

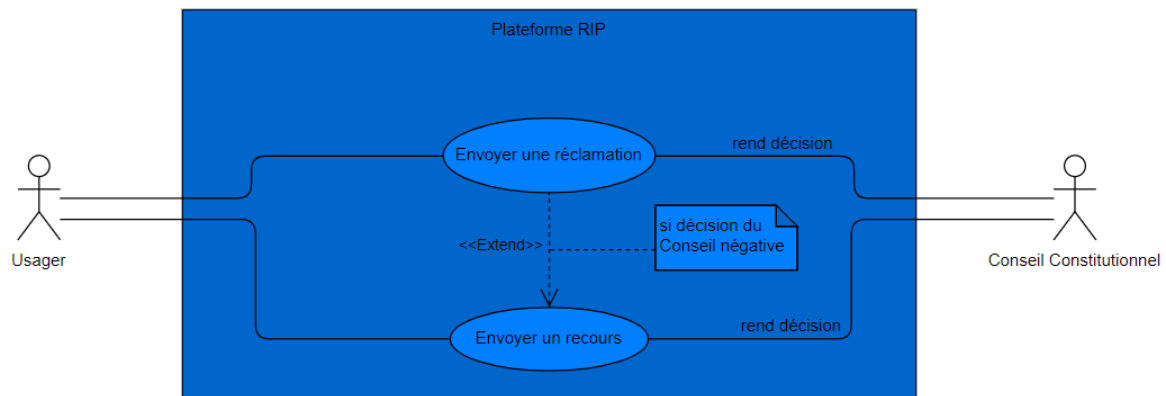
La fonctionnalité « **Gestion des soutiens** » est composée de plusieurs cas d'utilisations qui sont la **consultation du recueil de soutiens**, la **recherche dans le recueil de soutiens**, la **recherche par nom** et la **recherche par récépissé**, le **dépôt d'un soutien** et l'**ajout d'un soutien** au recueil.

#### Relations :

Tout d'abord n'importe quel usager pourra consulter ou effectuer une recherche dans le recueil mais seul un citoyen peut déposer un soutien. C'est le conseil Constitutionnel qui va vérifier les formulaires de dépôt de soutien. Nous avons mis une relation d'**inclusion** du cas d'utilisation « **rechercher un soutien dans le recueil** » vers le cas d'utilisation « **consulter le recueil** » car sur le site il faut cliquer sur l'onglet consulter les soutiens pour pouvoir accéder aux fonctionnalités de recherches. Nous avons également ajouté des relations de **généralisation** car la recherche dans le recueil peut être faite selon deux cas particuliers : par le nom ou par récépissé. Enfin nous avons ajouté une relation d'**extension** entre les fonctionnalités formulaire de dépôt de soutien et ajout du soutien au recueil car si le citoyen n'est pas inscrit sur les listes électorales, son soutien ne sera pas accepté et donc il ne sera pas ajouté au recueil de soutiens.

---

## DIAGRAMME DE 2<sup>EME</sup> NIVEAU REPRESENTANT LA FONCTIONNALITE GESTION DES RECLAMATIONS ET RECOURS



---

### EXPLICATIONS

#### Acteurs :

Nous retrouvons ici encore les mêmes acteurs que dans le diagramme de premier niveau.

#### Cas d'utilisation :

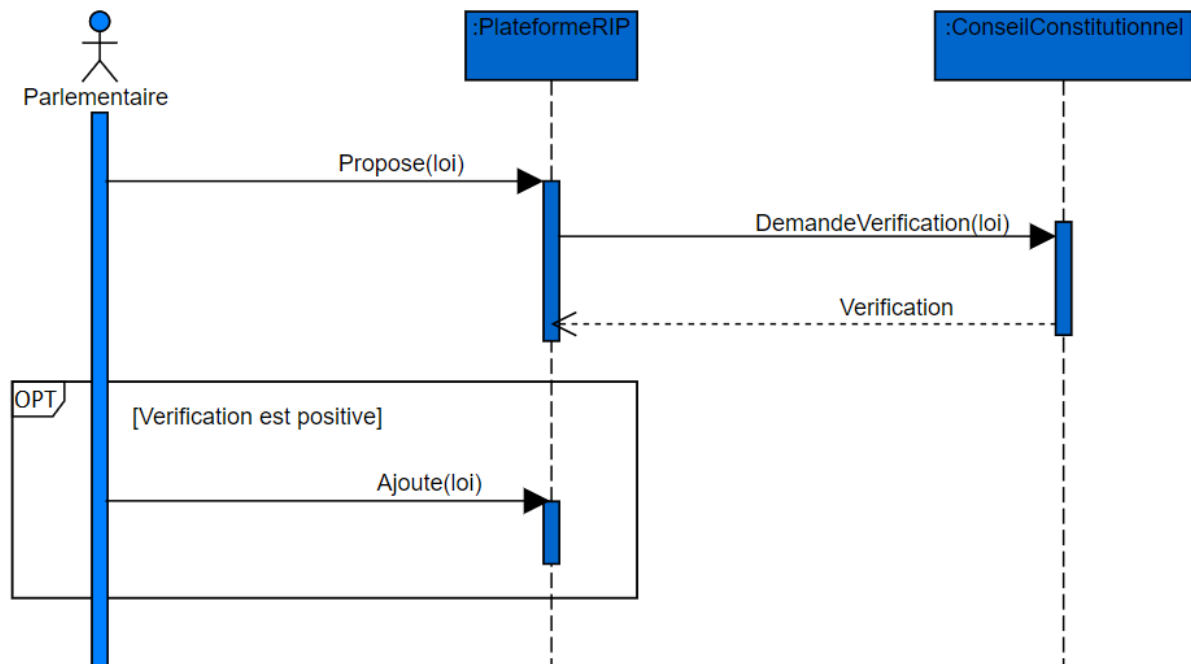
La fonctionnalité « [Gestion des réclamations et Recours](#) » ne contient que deux cas d'utilisations qui sont [envoyer une réclamation](#) et [envoyer un recours](#).

#### Relations :

L'utilisateur va pouvoir envoyer une réclamation ou un recours et ceux-ci seront traités par le Conseil Constitutionnel qui rendra sa décision. Nous avons mis une relation d'[extension](#) du cas d'utilisation envoyer une réclamation vers le cas d'utilisation envoyer un recours car on peut envoyer un recours seulement si la décision de la réclamation est négative. Le point d'extension serait donc après la réception de la décision du conseil.

---

## DIAGRAMME DE SÉQUENCE REPRÉSENTANT LA FONCTIONNALITÉ PROPOSITION DE LOI



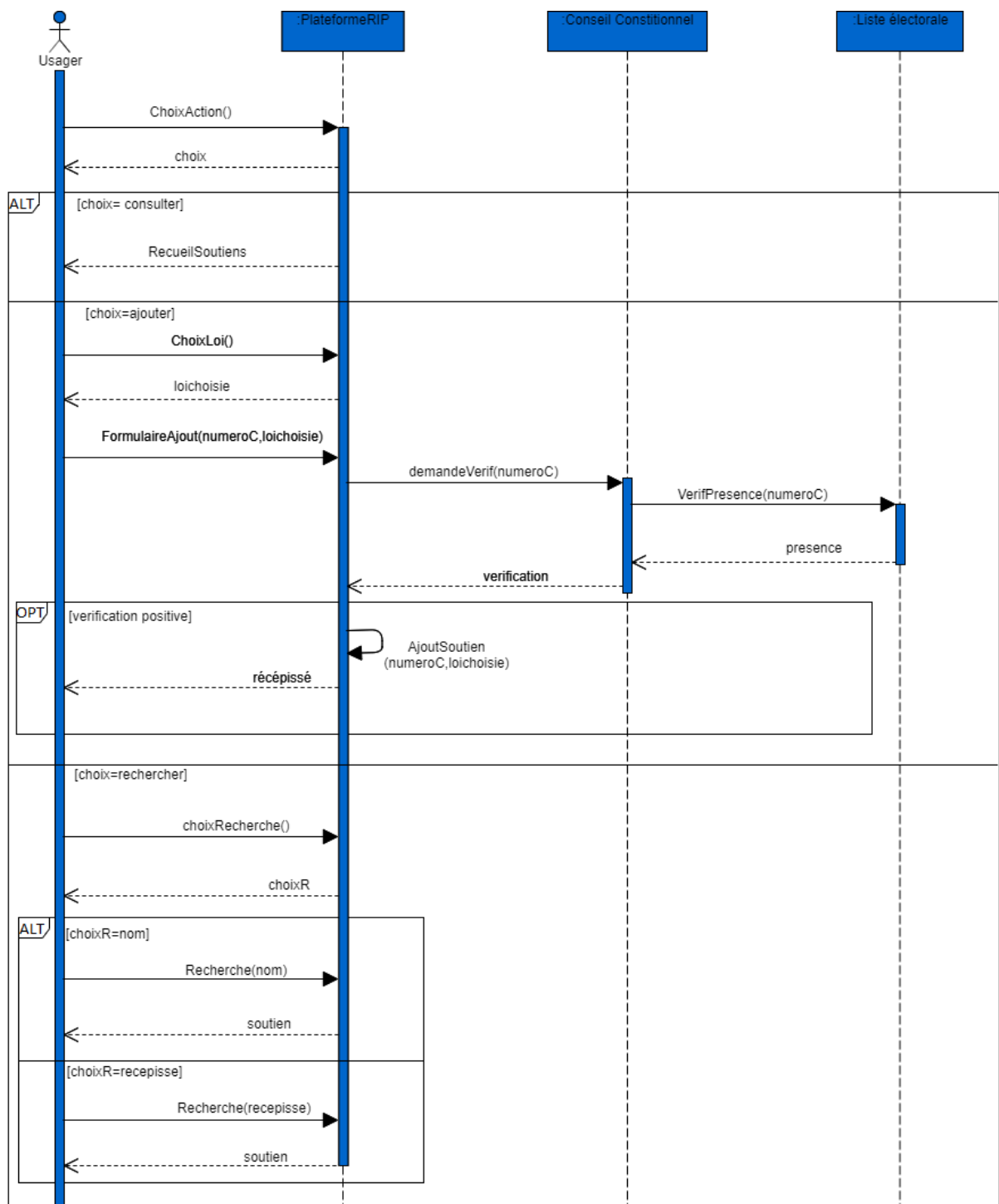
---

## EXPLICATIONS

On part du principe que le dépôt d'une proposition de loi se fait via la plateforme RIP.

Le parlementaire va donc envoyer à la plateforme sa proposition de loi, puis la plateforme va demander au Conseil Constitutionnel de vérifier cette proposition. Le Conseil Constitutionnel va renvoyer le résultat de sa vérification à la plateforme qui sera positif ou négatif. Puis on ajoute un fragment de séquence optionnel qui ne sera effectué que si la vérification du conseil était positive : si c'est le cas le parlementaire peut ajouter la proposition de loi à la liste des propositions de loi de la Plateforme.

## DIAGRAMME DE SÉQUENCE REPRÉSENTANT LA FONCTIONNALITÉ GESTION DES SOUTIENS





---

## EXPLICATIONS

On part du principe que l'utilisateur possède un numéro de citoyen qui va permettre au système de le reconnaître.

Tout d'abord l'Utilisateur va pouvoir choisir l'action qu'il veut effectuer parmi les différentes fonctionnalités proposées par la plateforme. On va ensuite avoir un fragment alternatif qui va dépendre de ce choix.

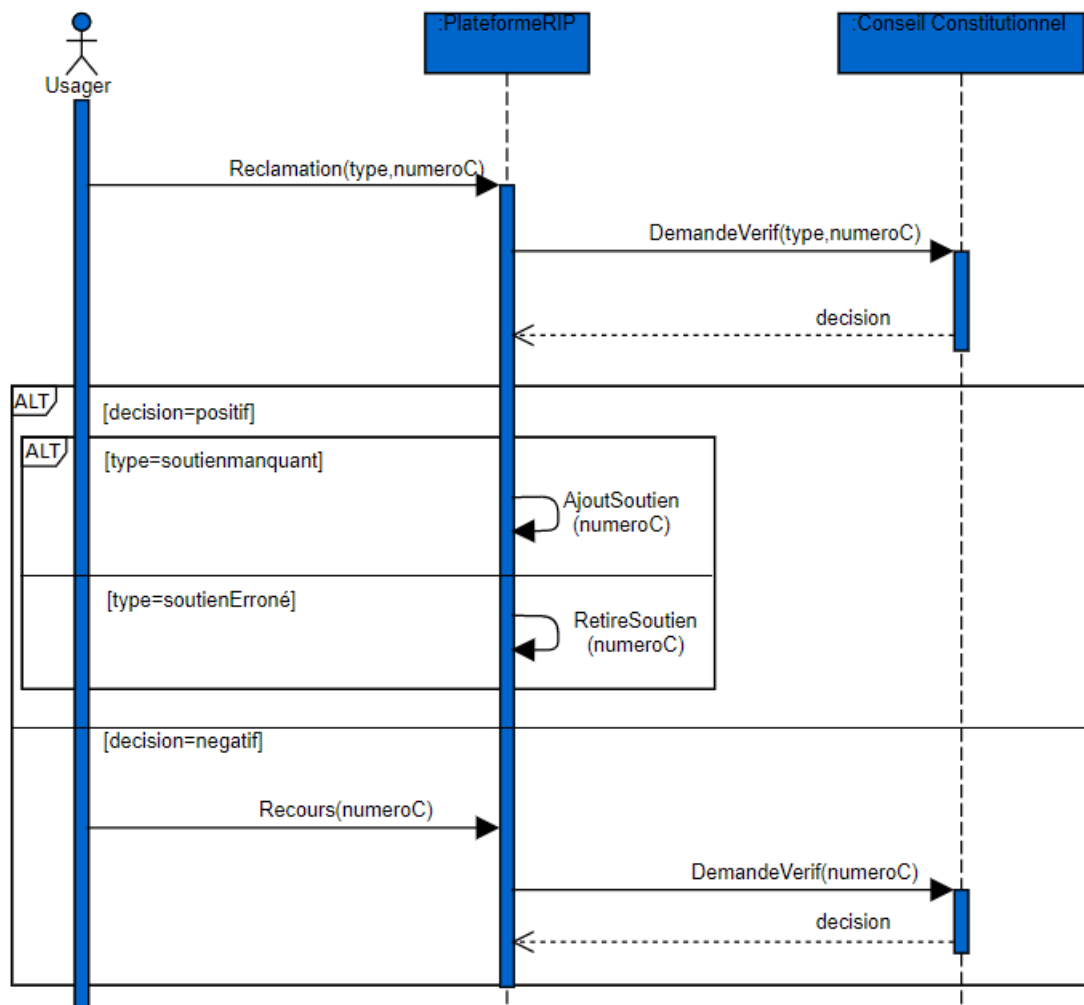
Si l'utilisateur a choisi de consulter le recueil alors la plateforme va lui renvoyer le recueil de soutiens.

Si l'utilisateur a choisi d'ajouter un soutien au recueil alors il va devoir tout d'abord choisir la proposition de loi parmi celles proposées sur la plateforme. Une fois cela fait il va pouvoir remplir le formulaire de dépôt dans lequel il indique son numéro de citoyen. Puis la plateforme va demander au conseil constitutionnel de vérifier si le citoyen apparaît ou non sur la liste électorale. Le conseil constitutionnel consulte la liste électorale et envoie le résultat de sa vérification. On a ensuite un fragment optionnel qui s'exécute si cette vérification est positive : le soutien est ajouté à la plateforme et un récépissé est envoyé à l'utilisateur pour lui indiquer que son soutien a bien été ajouté.

Si l'utilisateur a choisi de rechercher un soutien dans le recueil il va tout d'abord devoir choisir quel type de recherche il souhaite faire. On a donc à nouveau un fragment alternatif : si l'utilisateur a choisi la recherche par nom il devra envoyer le nom qu'il souhaite rechercher et la plateforme lui renverra le soutien correspondant (le soutien renvoyé peut être nul s'il n'existe pas de soutien à ce nom) et si l'utilisateur a choisi la recherche par récépissé il devra envoyer le récépissé qu'il souhaite et la plateforme lui renverra le soutien correspondant.

Nous ne l'avons pas mis mais nous aurions pu rajouter une grande boucle LOOP qui engloberait tout le diagramme car l'utilisateur pourra recommencer à utiliser ces fonctionnalités à l'infini.

## DIAGRAMME DE SÉQUENCE REPRÉSENTANT LA FONCTIONNALITÉ GESTION DES RÉCLAMATIONS ET RECOURS



## EXPLICATIONS

On part du principe que l'utilisateur qui va effectuer une réclamation ou un recours possède un numéro de citoyen qui permettra au conseil constitutionnel de le rechercher dans la liste électorale.

L'utilisateur va pouvoir envoyer une réclamation avec un type de réclamation et son numéro de citoyen. La plateforme va ensuite transmettre cette réclamation au conseil constitutionnel qui va rendre sa décision. On a ensuite plusieurs cas de figure donc on va mettre un fragment alternatif : si la décision du conseil est positive, on va regarder le type de réclamation dans un deuxième fragment alternatif : si la réclamation était que le soutien du citoyen n'apparaissait pas dans le recueil, la plateforme va ajouter ce soutien au recueil, si la réclamation était qu'un soutien apparaissait dans le recueil alors qu'il ne devrait pas, la plateforme va supprimer ce soutien (il y a sûrement d'autres types de réclamation que nous n'avons pas traité). En revanche si la décision du conseil est négative, l'utilisateur va pouvoir envoyer un recours qui sera transmis au Conseil Constitutionnel qui rendra sa décision.

---

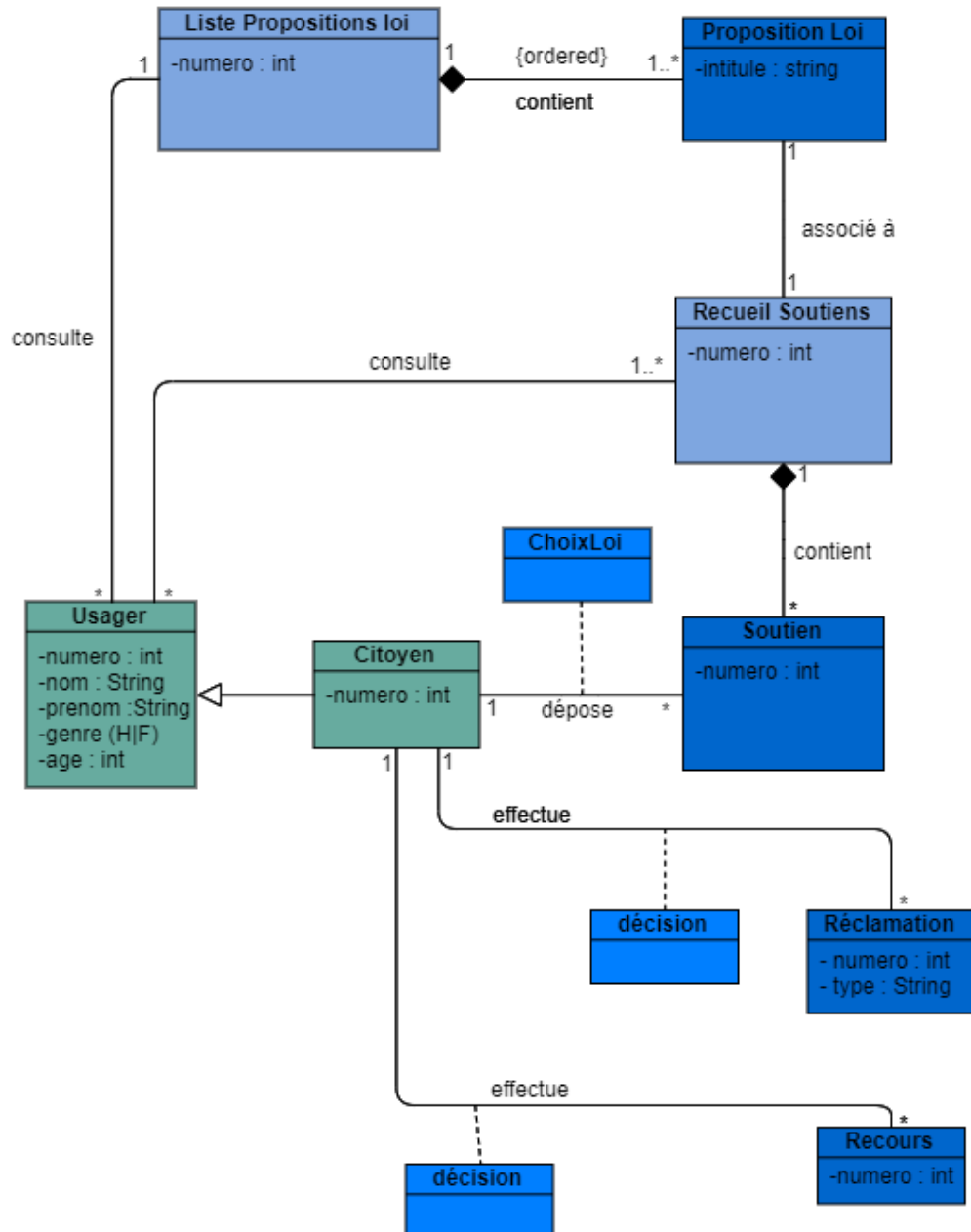
## CONCLUSION

Pour faire nos diagrammes Use Case nous avons commencé par essayer de repérer les différents éléments principaux : les acteurs et les différentes fonctionnalités offertes par la plateforme. Puis nous nous sommes intéressés aux relations entre ces différents éléments.

Ce qui a été compliqué pour nous a été de pouvoir représenter un maximum d'informations en utilisant un maximum des notions vues en cours pour avoir un travail assez complet tout en essayant de garder de la cohérence par rapport au sujet. Le principal défaut de nos diagrammes est qu'ils ne sont pas totalement fidèles à la réalité, notamment pour les propositions de loi qui dans la réalité sont sûrement traitées en dehors de la plateforme RIP. Nous n'avons également pas pu représenter toutes les informations et les subtilités du sujet.

## Description Structurelle

### DIAGRAMME DE CLASSE



---

## EXPLICATIONS

On retrouve donc l'utilisateur représenté sous forme d'une classe qui possède les attributs numéro (comme il peut y avoir des homonymes on l'utilisera comme identifiant), nom, prénom, genre et âge. L'utilisateur peut consulter la liste des propositions de loi (il n'y en a qu'une donc on met 1 en multiplicité) et consulter le recueil de soutiens (il peut consulter plusieurs recueils de soutiens associés chacun à une proposition de loi différente on met donc la multiplicité à 1..\* car il y aura au moins un recueil).

On retrouve également la classe Citoyen qui hérite de la classe usager car le citoyen est un usager particulier. Il possède en plus un attribut numéro. Le citoyen peut déposer 0 ou plusieurs soutiens (pour des loi différentes) pour la loi qu'il aura choisie. On ajoute donc une classe association pour le choix de cette loi. Le citoyen peut également effectuer 0 ou plusieurs réclamations et recours et une décision leur sera associée : on ajoute donc pour les deux cas une classe association décision.

La liste de proposition de loi est caractérisée par un attribut numéro et est une composition de propositions de loi qui contiendra 1 à plusieurs propositions de loi. On peut supposer que cette liste est ordonnée et on l'indique sur le diagramme par {ordered}.

Une proposition de loi est caractérisée par un intitulé et est associée à un recueil de soutiens caractérisé par un numéro qui lui-même est une composition de soutiens (nous avons mis la multiplicité à \* car une proposition de loi peut ne recueillir aucun ou plusieurs soutiens). La classe soutien est caractérisé par un attribut numéro.

Pour finir la classe réclamation est caractérisée par un attribut numéro et un attribut type. Et la classe recours par un attribut numéro. Une réclamation et un recours sont effectués par un seul citoyen.

Toutes les classes que nous avons représenté sont persistantes car il faut qu'à chaque fois que l'on accède à la plateforme, elle ait gardé en mémoire la liste des propositions de loi, le recueil des soutiens et les citoyens associés aux soutiens ainsi que les réclamations et les recours sans que ces informations ne soient remises à 0.

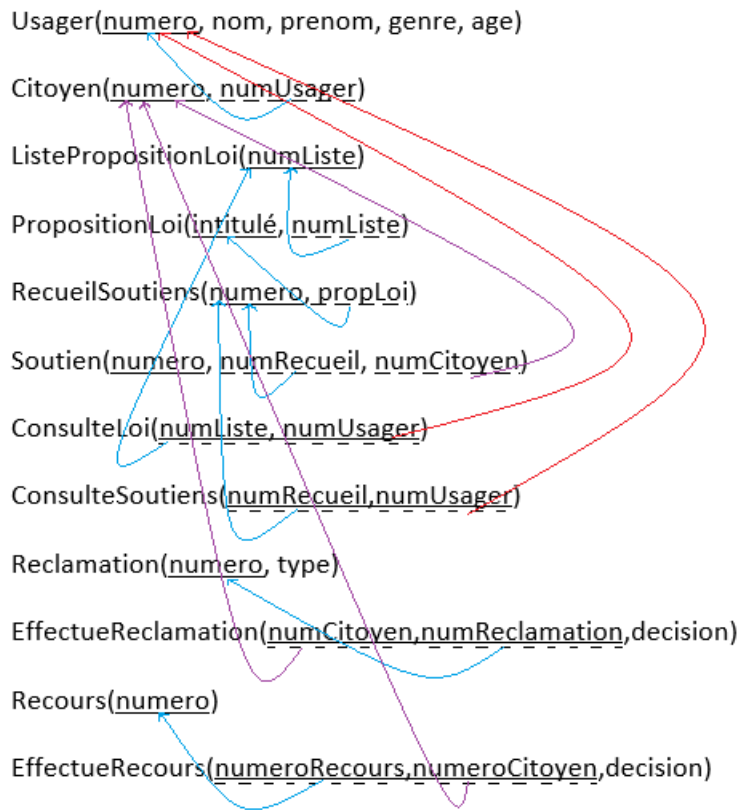
---

## CONCLUSION

Pour faire notre diagramme de classe nous avons essayé de ne garder que les principaux éléments de notre diagramme Use Case : nous n'avons gardé que les fonctionnalités et acteurs principaux que nous avons déclaré comme des classes. Ensuite nous avons ajouté les relations présentes entre ces différentes classes. Pour finir nous avons ajouté les multiplicités.

Le principal défaut de notre diagramme de classe est qu'il ne représente pas la globalité du système (la plateforme RIP) mais plutôt une sorte de résumé de celui-ci car nous n'avons pas pu inclure tous les éléments.

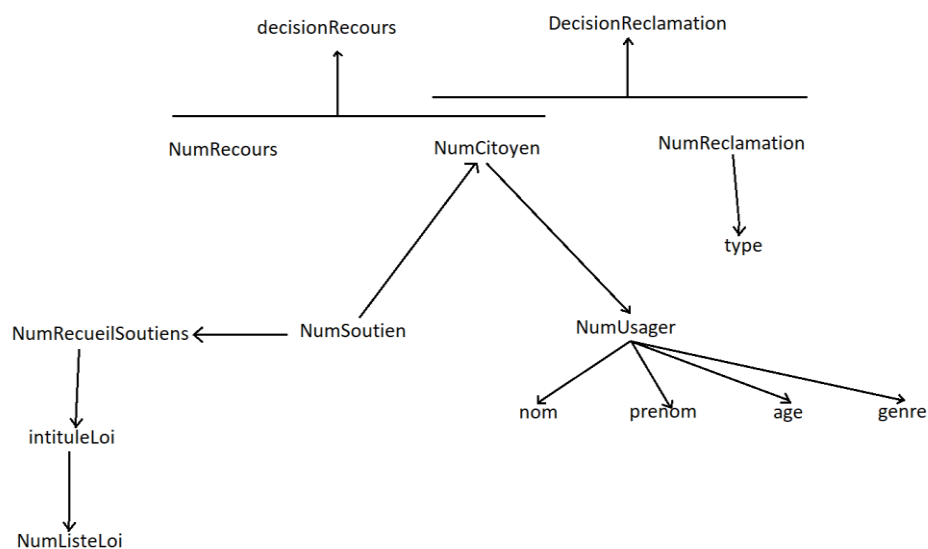
### Traduction dans le modèle relationnel



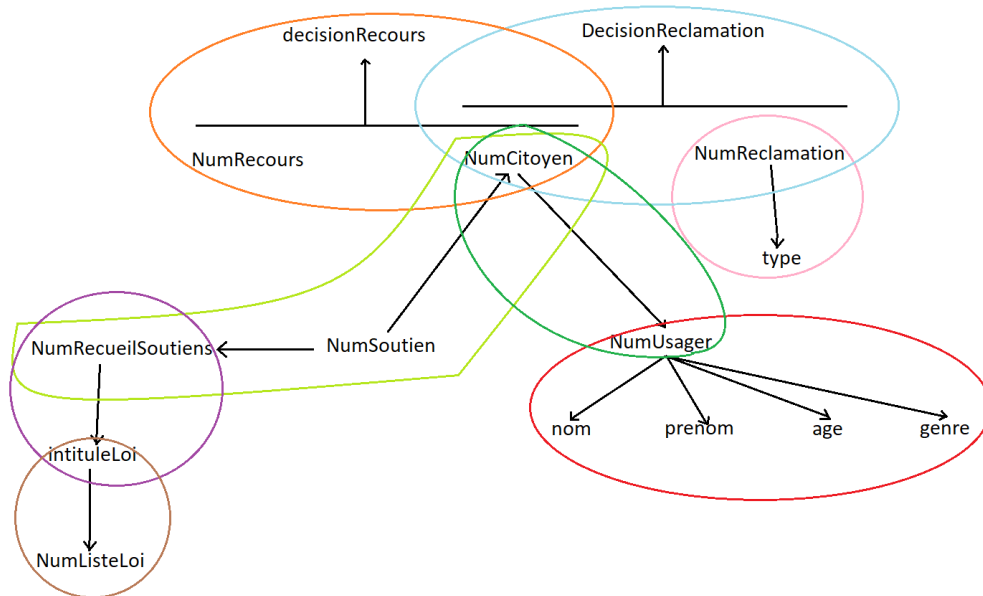
Nous avons fait apparaître les différents identifiants des relations qui apparaissent en souligné et également les identifiants externes qui apparaissent en souligné en pointillés.

### VERIFICATION FORMES NORMALES

Nous avons représenté nos différents attributs dans un schéma en faisant apparaître les différentes dépendances fonctionnelles entre eux.



Si on décompose en suivant les différentes sources de dépendances fonctionnelles on obtient nos relations :




1<sup>ère</sup> Forme normale : Nos relations sont bien en première forme normale car chaque valeur de chaque attribut de chaque tuple est une valeur simple.

2<sup>ème</sup> Forme normale : Nos relations sont bien en deuxième forme normale car chaque attribut qui ne fait pas partie d'un identifiant dépend d'un identifiant.

3<sup>ème</sup> Forme normale : Nos relations sont bien en 3<sup>ème</sup> forme normale car chaque attribut qui ne fait pas partie d'un identifiant dépend d'un identifiant directement.

Forme normale de Boyce-Codd : Nos relations sont bien en forme normale de Boyce-Codd car toute source de dépendance fonctionnelle est un identifiant.

## 2. PARTIE XML



The screenshot shows the ADPrip website header with navigation links: [adrip.fr](#), [signez la pétition](#), [Accueil](#), [Carte des soutiens](#), [Évolution des soutiens](#), [Palmarès](#), [Widget](#), and [Données brutes](#) (circled in red). The main content area features a blue background with the text: 'Projet de loi pour la tenue d'un référendum sur la privatisation, ou non, du groupe ADP', 'soutiens publiés par l'intérieur\* / soutiens nécessaires', '959 170 / 4 717 396', '(20,33%)', and 'soit au maximum environ 1 018 700 soutiens envoyés en tout'. A large blue banner on the right says 'SAUVONS ADP!'. Below the banner, a paragraph states: 'Pour réaliser nos différents documents XML nous avons utilisé les données brutes disponibles sur la plateforme ADPRip que l'on peut voir ci-dessus.'

### I. REPRÉSENTATION DES DONNÉES PAR COMMUNES

Voici un exemple de données brutes d'une commune récupérée sur la plateforme :

```
"41094":{"nom":"Fresnes","electeurs":959,"h_electeurs":16105,"homonymes":["89183","02333","21287","94034"],"soutiens":17,"h_soutiens":295}
```

On y trouve différentes informations : le code Commune, le nom, le nombre d'électeurs total, le nombre d'électeurs total des communes portant le même nom, la liste des codes des communes portant le même nom, le nombre de soutiens et enfin le nombre de soutiens des communes homonymes. Nous avons utilisé ces différentes informations pour proposer un document XML.



*Structures XML et DTD associée*

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE communes SYSTEM "communes.dtd">
<communes>
  <commune code="C21115" nom="Buncey" departement="Côte-d'or">
    <nbelecteurs>290</nbelecteurs>
    <nbsoutiens propositionloi="ADP">0</nbsoutiens>
  </commune>
  <commune code="C09056" nom="Bézac" departement="Ariège">
    <nbelecteurs>272</nbelecteurs>
    <nbsoutiens propositionloi="ADP">1</nbsoutiens>
  </commune>
  <commune code="C30244" nom="Saint-Clément">
    <nbelecteurs>267</nbelecteurs>
    <nbsoutiens propositionloi="ADP">3</nbsoutiens>
    <homonymes>
      <homonyme code="C07226"/>
      <homonyme code="C19194"/>
      <hnbelecteurs>4452</hnbelecteurs>
      <hnbsoutiens>51</hnbsoutiens>
    </homonymes>
  </commune>
  <commune code="C89474" nom="Villiers-Vineux" departement="Yonne">
    <nbelecteurs>231</nbelecteurs>
    <nbsoutiens propositionloi="ADP">2</nbsoutiens>
  </commune>
  <commune code="C07226" nom="Saint-Clément">
    <nbelecteurs>1180</nbelecteurs>
    <nbsoutiens propositionloi="ADP">22</nbsoutiens>
    <homonymes>
      <homonyme code="C30244"/>
      <homonyme code="C19194"/>
      <hnbelecteurs>4452</hnbelecteurs>
      <hnbsoutiens>51</hnbsoutiens>
    </homonymes>
  </commune>
  <commune code="C19194" nom="Saint-Clément">
    <nbelecteurs>3005</nbelecteurs>
    <nbsoutiens propositionloi="ADP">26</nbsoutiens>
    <homonymes>
      <homonyme code="C30244"/>
      <homonyme code="C07226"/>
      <hnbelecteurs>4452</hnbelecteurs>
      <hnbsoutiens>51</hnbsoutiens>
    </homonymes>
  </commune>
</communes>
```

Les deux premières lignes du document représentent le prologue : tout d'abord on fait la déclaration XML (indications pour le processeur de document : version et encodage utilisés) puis on déclare la DTD externe qui lui est associée (elle sera appelée lors de la validation du document).

```
<!ELEMENT communes (commune*)>
<!ELEMENT commune (nbelecteurs,nbsoutiens,homonymes?)>
<!ELEMENT nbelecteurs (#PCDATA)>
<!ELEMENT nbsoutiens (#PCDATA)>
<!ELEMENT homonymes (homonyme+,hnbelecteurs,hnbsoutiens)>
<!ELEMENT homonyme EMPTY>
<!ELEMENT hnbelecteurs (#PCDATA)>
<!ELEMENT hnbsoutiens (#PCDATA)>
<!ATTLIST commune code ID #REQUIRED
                  nom CDATA #REQUIRED
                  departement CDATA #IMPLIED>
<!ATTLIST nbsoutiens propositionloi CDATA #FIXED "ADP">
<!ATTLIST homonyme code IDREF #REQUIRED>
```

#### Les différents éléments :

Nous commençons par ouvrir une balise « *communes* » qui englobera toutes les balises « *commune* » qui représentent les communes de France (dans l'absolu car bien évidemment nous n'avons pu ici ne représenter qu'un petit échantillon de celles-ci). Dans la DTD nous déclarons donc un nouvel élément **communes** qui est un composé d'éléments **commune** avec l'indicateur d'occurrence **\*** qui indique qu'on aura un nombre quelconque d'éléments **commune**.

Chaque balise « *commune* » contient les balises « *nbelecteurs* », « *nbsoutiens* » et parfois « *homonymes* » s'il existe des communes portant le même nom, comme ce n'est pas obligatoire on va indiquer dans la DTD l'élément **homonymes** suivi de l'indicateur d'occurrence **?** qui signifie que l'élément pourra apparaître 0 ou 1 fois.

Ensuite les balises « *nbelecteurs* » et « *nbsoutiens* » ne contiendront pas d'autres balises mais juste du texte, on l'indique donc dans la DTD grâce à **#PCDATA** qui spécifie que les éléments **nbelecteurs** et **nbsoutiens** contiendront du texte simple.

La balise facultative « *homonymes* » va contenir les balises « *homonyme* » « *hnbelecteurs* » et « *hnbsoutiens* ». Si la balise est présente c'est qu'il existe au moins 1 homonyme pour la commune, on l'indique donc dans la DTD en déclarant l'élément **homonyme** suivi de l'indicateur d'occurrence **+** qui signifie que l'élément apparaît 1 ou plusieurs fois. Les balises « *hnbelecteurs* » et « *hnbsoutiens* » ne contiendront pas d'autres balises mais juste du texte, on l'indique donc dans la DTD comme vu précédemment grâce à **#PCDATA**. La balise « *homonyme* » quant à elle ne contiendra rien, on la déclare donc comme un élément vide dans la DTD grâce à **EMPTY**.

#### Les différents attributs des éléments :

Chaque balise « *commune* » contient les attributs *code* et *nom* et parfois *departement* (comme cette information n'était pas présente dans les données brutes nous avons décidé de ne pas la rendre obligatoire). Comme le code de chaque commune est unique on le déclare comme attribut de type identifiant dans la DTD avec **ID** et on le rend obligatoire avec **#REQUIRED**. Le nom de la commune est quant à lui de type texte simple on l'indique donc dans la DTD par **CDATA** et on le rend également obligatoire grâce à **#REQUIRED**. Pour finir l'attribut *departement* est de type texte simple **CDATA** et comme on a décidé de le rendre facultatif on l'indique par **#IMPLIED** dans la DTD.

Chaque balise « *nbsoutiens* » possède un attribut *propositionloi* (car un soutien est lié à une proposition de loi en particulier). Comme pour le moment il n'y a qu'une seule proposition de loi disponible on décide donc de donner une valeur fixe à l'attribut qui sera "ADP". On l'indique dans la DTD par **#FIXED "ADP"** qui permet de définir une constante pré assignée.

Pour finir chaque balise « homonyme » contient un code qui est un identifiant externe sur la balise « commune » qui lui est associée. On indique cela par **IDREF** dans la DTD et on le rend également obligatoire avec **#REQUIRED**.

Pour vérifier que notre document xml est bien valide avec notre DTD, on lance la commande suivante dans un terminal : `xmllint --noout --valid communes.xml`. Celle-ci ne nous renvoie pas d'erreur. On peut aussi utiliser le site <http://validator.aborla.net> qui est un correcteur XML en ligne pour vérifier que notre document est bien formé.

### Requêtes XPath

Pour exécuter nos requêtes XPath sur notre document nous devons lancer la commande suivante dans un terminal : `xmllint --shell communes.xml`. On pourra ensuite écrire nos requêtes directement précédées par la commande `cat`.

- Afficher le nom des communes ayant au moins une commune homonyme

Pour faire cela on récupère les communes qui correspondent à la condition que son nombre de balise homonyme soit supérieur à 1, puis on affiche son attribut nom.

```
/ > cat /communes/commune[count(homonymes/homonyme)>1]/@nom
-----
nom="Saint-Clément"
-----
nom="Saint-Clément"
-----
nom="Saint-Clément"
```

Ce résultat est logique car dans notre document nous avons 3 communes homonymes au nom de Saint-Clément.

- Afficher le code des communes ayant un pourcentage de soutien supérieur à 1%

Pour faire cela on récupère les communes qui remplissent la condition que la valeur contenue dans *nbsoutiens* multipliée par 100 divisée par la valeur contenue dans *nbelecteurs* (calcul du pourcentage de soutiens) soit supérieur à 1, puis on affiche son attribut code.

```
/ > cat /communes/commune[((nbsoutiens*100)div(nbelecteurs))>1]/@code
-----
code="C30244"
-----
code="C07226"
```

En effet dans notre document Xml si on calcule le pourcentage pour la commune au code C30244 on obtient environ :  $3 \times 100 / 267 = 1,12\%$  et pour la commune au code C07226 :  $22 \times 100 / 1180 = 1,86\%$ .

---

## II. REPRÉSENTATION DES DONNÉES PAR DÉPARTEMENTS

Voici un exemple de données brutes d'un département récupéré sur la plateforme :

```
"24":{"nom":"Dordogne","electeurs":309179,"soutiens":2991}
```

On y trouve différentes informations : Le numéro du département, son nom, son nombre d'électeurs et son nombre de soutiens. Nous avons utilisé ces différentes informations pour proposer un document XML.

### *Structures XML et DTD associée*

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE departements SYSTEM "departements.dtd">
<departements pays="france">
  <departement numero=" 87" nom="Haute-Vienne" type="metropolitain">
    <nbelecteurs>260823</nbelecteurs>
    <nbsoutiens propositionloi="ADP">3873</nbsoutiens>
  </departement>
  <departement numero=" 24" nom="Dordogne" type="metropolitain">
    <nbelecteurs>309179</nbelecteurs>
    <nbsoutiens propositionloi="ADP">4482</nbsoutiens>
  </departement>
  <departement numero=" 15" nom="Cantal" type="metropolitain">
    <nbelecteurs>114481</nbelecteurs>
    <nbsoutiens propositionloi="ADP">1174</nbsoutiens>
  </departement>
  <departement numero=" 01" nom="Ain" type="metropolitain">
    <nbelecteurs>413702</nbelecteurs>
    <nbsoutiens propositionloi="ADP">5288</nbsoutiens>
  </departement>
  <departement numero=" 11" nom="Aude" type="metropolitain">
    <nbelecteurs>269293</nbelecteurs>
    <nbsoutiens propositionloi="ADP">4645</nbsoutiens>
  </departement>
  <departement numero=" 10" nom="Aube" type="metropolitain">
    <nbelecteurs>201439</nbelecteurs>
    <nbsoutiens propositionloi="ADP">1919</nbsoutiens>
  </departement>
  <departement numero=" 986" nom="WallisEtFutuna" type="dom-tom">
    <nbelecteurs>8756</nbelecteurs>
    <nbsoutiens propositionloi="ADP">6</nbsoutiens>
  </departement>
  <departement numero=" 75" nom="Paris" type="metropolitain">
    <nbelecteurs>1306831</nbelecteurs>
    <nbsoutiens propositionloi="ADP">62938</nbsoutiens>
  </departement>
  <departement numero=" 69" nom="Rhône" type="metropolitain">
    <nbelecteurs>1114503</nbelecteurs>
    <nbsoutiens propositionloi="ADP">22866</nbsoutiens>
  </departement>
  <departement numero=" 976" nom="Mayotte" type="dom-tom">
    <nbelecteurs>78076</nbelecteurs>
    <nbsoutiens propositionloi="ADP">252</nbsoutiens>
  </departement>
</departements>
```

```

</departement>
<departement numero="_17" nom="Charente-Maritime" type="metropolitain">
  <nbelecteurs>490697</nbelecteurs>
  <nbsoutiens propositionloi="ADP">7004</nbsoutiens>
</departement>
<departement numero="_971" nom="Guadeloupe" type="dom-tom">
  <nbelecteurs>307450</nbelecteurs>
  <nbsoutiens propositionloi="ADP">1116</nbsoutiens>
</departement>
<departement numero="_977" nom="StBarthélemy" type="dom-tom">
  <nbelecteurs>5177</nbelecteurs>
  <nbsoutiens propositionloi="ADP">40</nbsoutiens>
</departement>
<departement numero="_52" nom="Haute-Marne" type="metropolitain">
  <nbelecteurs>130980</nbelecteurs>
  <nbsoutiens propositionloi="ADP">1047</nbsoutiens>
</departement>
<departement numero="_978" nom="StMartin" type="dom-tom">
  <nbelecteurs>18150</nbelecteurs>
  <nbsoutiens propositionloi="ADP">74</nbsoutiens>
</departement>
<departement numero="_07" nom="Ardèche" type="metropolitain">
  <nbelecteurs>247371</nbelecteurs>
  <nbsoutiens propositionloi="ADP">4691</nbsoutiens>
</departement>
</departements>

```

Tout comme dans notre premier document on commence par déclarer le prologue ainsi que la DTD externe associée .

```

<!ELEMENT departements (departement*)>
<!ELEMENT departement (nbelecteurs,nbsoutiens)>
<!ELEMENT nbelecteurs (#PCDATA)>
<!ELEMENT nbsoutiens (#PCDATA)>
<!ATTLIST departements pays CDATA #REQUIRED>
<!ATTLIST departement numero ID #REQUIRED
                  nom CDATA #REQUIRED
                  type (dom-tom|metropolitain) #REQUIRED>
<!ATTLIST nbsoutiens propositionloi CDATA #FIXED "ADP">

```

#### Les différents éléments :

Nous commençons par ouvrir une balise « *departements* » qui englobera toutes les balises « *departement* » qui représentent les départements de France (dans l'absolu car bien évidemment nous n'avons pu ici ne représenter qu'un petit échantillon de ceux-ci). Dans la DTD nous déclarons donc un nouvel élément **départements** qui est un composé d'éléments **département** avec l'indicateur d'occurrence **\*** qui indique qu'on aura un nombre quelconque d'éléments **département**.

Chaque balise « *departement* » contient les balises « *nbelecteurs* », « *nbsoutiens* » on déclare donc ces éléments dans la DTD. Les éléments **nbelecteurs** et **nbsoutiens** contiendront du texte, on l'indique donc grâce à **#PCDATA** dans la DTD qui spécifie que le contenu de ces éléments est de type texte simple.

#### Les différents attributs des éléments :

Chaque balise « *departement* » contient les attributs *numero*, *nom* et *type* (cette information n'était pas présente dans les données brutes mais nous avons décidé de la rajouter pour enrichir notre document). Comme le numéro de chaque département est unique on le déclare comme attribut de type identifiant dans la DTD avec **ID** et on le rend obligatoire avec **#REQUIRED**. Le nom du département est quant à lui de type texte simple on l'indique donc dans la DTD par **CDATA** et on le rend également obligatoire grâce à **#REQUIRED**. Pour finir l'attribut *type* est de type énuméré qui peut prendre soit la valeur "metropolitain " soit la valeur "dom-tom " on l'indique dans la DTD sous la forme **(dom-tom|metropolitain)**, nous avons également décidé de le rendre obligatoire alors on l'indique par **#REQUIRED** dans la DTD.

Pour finir chaque balise « *nbsoutiens* » possède un attribut *propositionloi* pour les mêmes raisons que notre premier document XML. Comme pour le moment il n'y a qu'une seule proposition de loi disponible on décide donc de donner une valeur fixe à l'attribut qui sera "ADP". On l'indique dans la DTD par `#FIXED " ADP "` qui permet de définir une constante pré assignée.

Comme précédemment on vérifie que notre document est valide et bien formé (commande `xmllint --noout --valid departements.xml` et site <http://validator.aborla.net>).

### Requêtes XPath

Pour exécuter nos requêtes XPath sur notre document nous devons lancer la commande suivante dans un terminal : `xmllint --shell departements.xml`. On pourra ensuite écrire nos requêtes directement précédées par la commande `cat`.

- Afficher le nom des départements de type dom-tom

Pour faire cela on récupère les départements qui correspondent à la condition que leur attribut type soit égal à "dom-tom", puis on affiche leur attribut nom.

```
/ > cat /departements/departement[@type="dom-tom"]/@nom
-----
nom="WallisEtFutuna"
-----
nom="Mayotte"
-----
nom="Guadeloupe"
-----
nom="StBarthélemy"
-----
nom="StMartin"
```

Ce résultat est cohérent avec les départements contenus dans notre document XML.

- Afficher le nom des départements ayant plus de 4000 soutiens

Pour faire cela on récupère les départements qui correspondent à la conditions que la valeur contenue dans nbsoutiens soit supérieure à 4000, puis on affiche leur nom.

```
/ > cat /departements/departement[nbsoutiens>4000]/@nom
-----
nom="Dordogne"
-----
nom="Ain"
-----
nom="Aude"
-----
nom="Paris"
-----
nom="Rhône"
-----
nom="Charente-Maritime"
-----
nom="Ardèche"
```

En effet cela est cohérent avec notre document car si on vérifie on voit que la Dordogne a 4482 soutiens, l'Ain en a 5288 etc.

## REPRÉSENTATION D'UN FORMULAIRE DE DÉPÔT DE SOUTIEN

Pour ce dernier document xml nous nous sommes inspirés du formulaire de dépôt de soutien disponible sur la plateforme RIP présenté ci-dessous :

A quel endroit votez-vous ? \* ☒ en France ☐ à l'étranger

Département/collectivité d'outre-mer de vote \*

Recherchez votre commune de vote (exemple : sainte-cécile-les-vignes) \*

Renseignez le nom de la commune dans laquelle vous êtes inscrit pour voter et non son code postal ou son code Insee. Sélectionnez-la ensuite dans la liste proposée.

Quel est votre nom ? \*

Saisissez votre nom de naissance

☐ Je n'ai pas de prénom

Quels sont vos prénoms ?

Saisissez tous vos prénoms tels qu'inscrits sur votre carte électorale en les séparant par des espaces. N'utilisez pas de virgules dans ce champ.

Quel est votre sexe ? \* ☐ Féminin ☐ Masculin

Date naissance \*

### Structures XML et DTD associée

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE formulaires SYSTEM "formulaire.dtd">
<formulaires>
  <formulaire>
    <localisation type="france"/>
    <departement>21</departement>
    <commune>Dijon</commune>
    <nom>Dupont</nom>
    <prenom>Jean</prenom>
    <sexe type="Masculin" />
    <dateNaissance jour="12" mois="8" annee="1970"/>
    <paysNaissance>France</paysNaissance>
    <departementNaissance>21</departementNaissance>
    <communeNaissance>Beaune</communeNaissance>
  </formulaire>
  <formulaire>
    <localisation type="france"/>
    <departement>40</departement>
    <commune>Dax</commune>
    <nom>Dupont</nom>
    <prenom>Marie</prenom>
    <sexe type="Feminin" />
    <dateNaissance jour="12" mois="6" annee="1976"/>
    <paysNaissance>France</paysNaissance>
    <departementNaissance>64</departementNaissance>
    <communeNaissance>Bayonne</communeNaissance>
  </formulaire>
</formulaires>
```



Tout comme dans notre premier document on commence par déclarer le prologue ainsi que la DTD externe associée.

```
<!ELEMENT formulaires (formulaire*)>
<!ELEMENT formulaire (localisation,departement,commune,nom,prenom,sexe,
dateNaissance,paysNaissance,departementNaissance,communeNaissance)>
<!ELEMENT localisation EMPTY>
<!ELEMENT departement (#PCDATA)>
<!ELEMENT commune (#PCDATA)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA) >
<!ELEMENT sexe EMPTY>
<!ELEMENT dateNaissance EMPTY>
<!ELEMENT paysNaissance (#PCDATA)>
<!ELEMENT departementNaissance (#PCDATA)>
<!ELEMENT communeNaissance (#PCDATA)>
<!ATTLIST dateNaissance jour CDATA #REQUIRED
                        mois CDATA #REQUIRED
                        annee CDATA #REQUIRED >
<!ATTLIST localisation type (france|etranger) #REQUIRED>
<!ATTLIST sexe type (Feminin|Masculin) #REQUIRED>
```

#### Les différents éléments :

Nous commençons par ouvrir une balise « *formulaires* » qui englobera toutes les balises « *formulaire* » qui représentent, dans l'absolu, tous les formulaires de dépôt de soutien qui ont été remplis. Dans la DTD nous déclarons donc un nouvel élément **formulaires** qui est un composé d'éléments **formulaire** avec l'indicateur d'occurrence **\*** qui indique qu'on aura un nombre quelconque d'éléments **departement** car il peut n'y avoir aucun ou plusieurs formulaires remplis.

Nous avons ensuite une balise « *formulaire* » qui est un composé d'éléments : **localisation**, **departement**, **commune**, **nom**, **prenom**, **sexe**, **dateNaissance**, **paysNaissance**, **departementNaissance** et **communeNaissance** qui correspondent aux différents champs à remplir dans le formulaire de la plateforme. Les éléments **localisation**, **sexe** et **dateNaissance** auront un contenu vide donc on l'indique dans la DTD par **EMPTY**. Le reste des éléments quant à eux contiendront du texte on l'indique donc dans la DTD par **#PCDATA**.

#### Les différents attributs des éléments :

Chaque balise « *dateNaissance* » contient les attributs **jour**, **mois** et **annee**, ils sont tous trois obligatoires et de type texte simple donc on l'indique dans la DTD avec **#REQUIRED** et **CDATA**.

Chaque balise « *localisation* » contient un attribut **type** de type énuméré qui peut prendre les valeurs "france" ou "etranger" et il est obligatoire donc on ajoute **#REQUIRED** dans la DTD.

Pour finir, chaque balise « *sexe* » contient un attribut **type** de type énuméré qui peut prendre les valeurs "Feminin" ou "Masculin" et il est obligatoire donc on ajoute **#REQUIRED** dans la DTD.



## Requêtes XPath

- Afficher le prénom des personnes localisées en France

Pour faire cela on recherche les formulaires dont la localisation répond à la condition que son attribut type soit égal à la valeur "france". Ensuite on récupère les prénoms correspondants à ces formulaires.

```
/ > cat formulaires/formulaire/localisation[@type="france"]/../prenom
-----
<prenom>Jean</prenom>
-----
<prenom>Marie</prenom>
```

Cela est cohérent car les deux formulaires de notre document xml concernent des personnes localisées en France.

- Afficher le prénom des personnes de plus de 30 ans

Pour faire cela on récupère les formulaires dont la date de naissance correspond à la condition que son attribut année soustrait à 2019 (cela nous donne l'âge) soit supérieur à 30. Ensuite on récupère les prénoms correspondants à ces formulaires.

```
/ > cat /formulaires/formulaire/dateNaissance[(2019-@annee)>30]/../prenom
-----
<prenom>Jean</prenom>
-----
<prenom>Marie</prenom>
```

Cela est cohérent car les deux formulaires de notre document xml concernent des personnes nées dans les années 70 donc qui ont par conséquent plus de 30 ans.