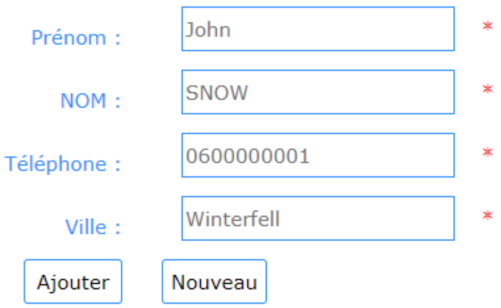
**R4.01 : TP AJAX/Spring**

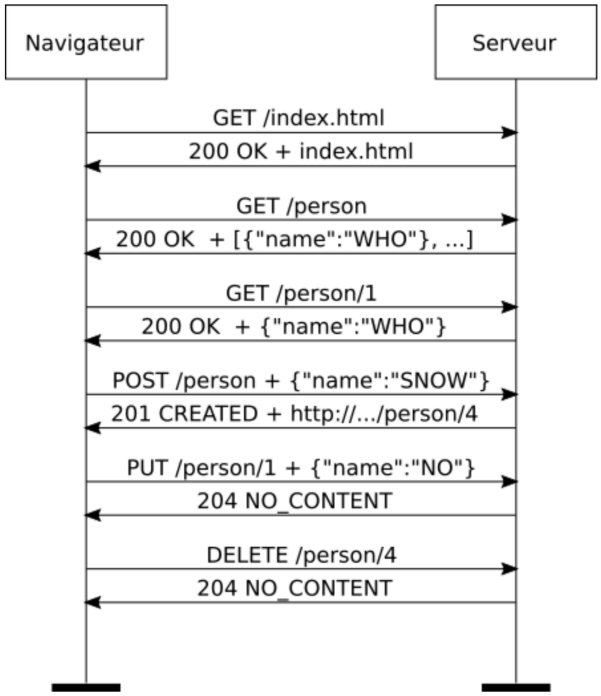



Une feuille de styles fournie (ci-dessous) place automatiquement les éléments d'un formulaire (`#idForm`) sur deux colonnes. Le premier élément (prénom) est placé sur la colonne de gauche, le second élément (zone de saisie + étoile) est placé sur la colonne de droite et ainsi de suite. Ce qui apparaît dans les zones de saisie est un exemple de ce qu'on peut y saisir (attribut `placeholder`).

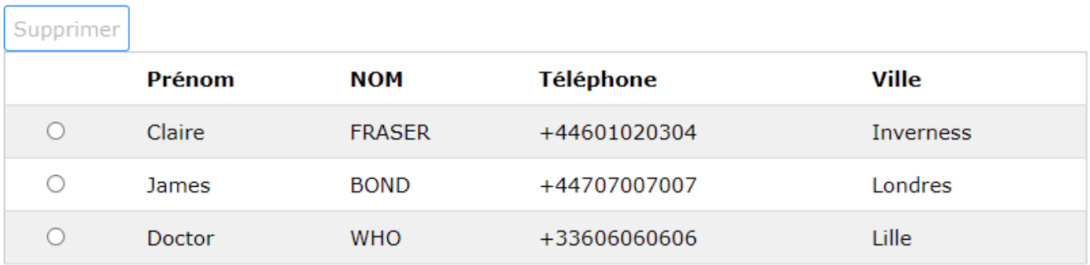
Le bouton Ajouter demande à l'API d'ajouter un contact dans l'annuaire. Le bouton Nouveau vide les zones de saisie.

La figure suivante montre les échanges de requêtes/réponses HTTP possibles grâce à cette IHM.

```
body * {
	margin-left: 1em;
}
#idForm {
	color: dodgerblue;
	display: grid;
	grid-template-columns: auto auto;
	gap: 1em 1em;
	grid-auto-rows: 2.2em;
	grid-auto-columns: max-content;
	margin: 2em;
	max-width: 60em;
}
#idForm>*:nth-child(odd) {
	place-self: end end;
}
#idForm>*:nth-child(even) {
	place-self: end start;
}
input {
	border: 1px solid dodgerblue;
	line-height: 2em;
}
button {
	background-color: #fff;
	border: 1px solid dodgerblue;
	border-radius: 3px;
	line-height: 2em;
	min-width: 5rem;
}
```




Renommer `index.html` en `tp.html` ;
Dans `index.html`, mettre en place le code HTML de cette partie de l'IHM.
Une seconde feuille de styles w3.css fournit la classe `w3-table-all` pour donner à un élément `table` l'aspect suivant :



Cette seconde partie de l'IHM (le tableau) présente le contenu de l'annuaire obtenu via l'API :

- La sélection d'un élément alimente le formulaire avec les informations de la ligne et transforme le bouton Ajouter en Modifier.
- Après modification d'au moins une zone de saisie, une action sur ce bouton demande à l'API de modifier le contact de l'annuaire. Un ajout ou une modification d'un contact met à jour cette partie de l'IHM.
- Une action sur Nouveau vide les zones de saisie et redonne à l'autre bouton son texte Ajouter et le comportement associé.
- Une action sur le bouton Supprimer demande à l'API de supprimer le contact de l'annuaire. En cas de succès, la ligne correspondante du tableau est supprimée.

Pour vous aider...

## Requête GET /entree

- Créer le fichier `js/app.js` ;
- le référencer dans `index.html` ;
- intégrer le comportement suivant :
- au chargement de la page, demander à l'API le contenu complet de l'annuaire ;
- quand cette réponse arrive, manipuler le DOM pour alimenter le tableau :
  - créer pour cela une classe `ContactsTable` qui réifie le tableau HTML ;
  - l'instancier au chargement de la page ;
  - prévoir une méthode `insert` qui insère une ligne dans le tableau ;
  - éviter les réplications (plusieurs lignes de même id) ;
- tester.

## Requête POST /entree

Lors du clic sur le bouton Ajouter,

- vérifier que les zones de saisie du formulaire sont non vides ;
  - créer pour cela une classe `ContactsForm` qui réifie le formulaire + bouton Ajouter ;
  - l'instancier au chargement de la page ;
  - dans le constructeur, installer le handler du clic sur Ajouter ;
  - prévoir une méthode `getPerson()` qui retourne un objet personne sans id ou `undefined` si une zone de saisie est vide ;
  - appeler `getPerson` sur clic du bouton Ajouter ;
- demander à l'API de créer une nouvelle ressource à partir des informations du formulaire ;
- quand la réponse arrive, refléter dans le tableau la création réalisée par l'API :
  - compléter l'objet personne émis sans id avec l'id renvoyé par la requête ;
- tester.

## Requête GET /entree/id

- Lors du clic d'un bouton radio (colonne de gauche), redemander à l'API la ressource personne de la ligne sélectionnée ;

    - pour cela, compléter le constructeur de `ContactsTable` ;
    - installer un handler unique sur l'objet `table` ;
    - utiliser `event.target` pour discriminer le bouton radio cliqué ;
- Quand la réponse arrive,
    - remplir le formulaire avec les informations de la ressource demandée y compris son id même s'il n'est pas visible ;
    - changer le libellé du bouton Ajouter en Modifier ;
    - pour cela ajouter à `ContactsForm` une méthode `update(person)` ;
- tester.

## Requête PUT /entree/id

- Lors du clic du bouton Modifier, demander à l'API de mettre à jour la ressource personne à partir des informations du formulaire ;
- pour cela, modifier `getPerson()` et renvoyer :
    - un objet personne avec id en cas de modification ;
    - un objet personne sans id en cas d'ajout ;
    - `undefined` si une zone de saisie est non vide ;
- quand la réponse arrive, refléter dans le tableau la mise à jour réalisée par l'API ;
    - pour cela, distinguer dans `insert` le cas où la ressource modifiée est déjà présente dans le tableau ;
    - dans ce cas la mettre à jour ;
    - renommer `insert` en `upsert` (contraction de update et insert) ;
- tester.

## Effacement du formulaire

- Lors du clic du bouton Nouveau :
    - installer un handler pour le bouton dans le constructeur de `ContactsForm` ;
    - vider les zones de saisie avec un appel à un `update()` modifié sans l'argument `person` ;
    - remettre le formulaire en état de créer une nouvelle ressource (plutôt que de la modifier) ;
- tester.

## Requête DELETE /entree/id

- Rendre d'abord actif le bouton Supprimer dès qu'un bouton radio est `checked` ;
    - pour cela, compléter le handler installé sur l'élément `table` ;
    - utiliser la propriété `disabled` du bouton Supprimer ;
- lors du clic du bouton Supprimer, demander à l'API de supprimer la ligne du bouton radio `checked` ;
    - installer le handler dans le constructeur de `ContactsTable` ;
- Quand la réponse arrive, supprimer la ligne correspondante dans le tableau ;
    - pour cela, ajouter une méthode `remove(id)` dans ContactsTable ;
    - en l'absence de bouton radio `checked`, désactiver aussi le bouton Supprimer ;
    - vider les zones de saisie ;
    - remettre le formulaire en état de créer une nouvelle ressource (plutôt que de la modifier) ;
- tester.