TRAVAIL PRATIQUE #2 – [12,5% de la note finale]

Validation de formulaire – Côté client + Modèle MVC

Objectifs du travail

- Approfondir votre maîtrise du processus de validation de formulaires HTML;
- Tirer profit de la validation intégrée de Bootstrap (version 5.3);
- Effectuer des validations personnalisées supplémentaires en utilisant notamment des expressions régulières simples;
- Mettre en application le patron de conception MVC dans une application web;
- Organiser son code de manière structurée et lisible.

Mise en contexte

Nous reprenons ici notre patron MVC présenté dans la série d'exercices du cours 6, mais cette fois-ci, notre application permet de gérer des employés d'une compagnie plutôt que des produits.

Pour le **modèle**, nous définirons nos employés à l'aide des attributs suivants :

- Matricule;
- Prénom
- Nom;
- Salaire;
- Département;
- Courriel;
- Retraité(e);
- Date de retraite.

Vous disposez également du fichier *Departements.js* qui contient un tableau (invariable) contenant la liste des départements de la compagnie :

```
// Tableau des départements (à importer dans le modèle)
const _departements = [
    { id: 1, nom: 'Comptabilité' },
    { id: 2, nom: 'Finances' },
    { id: 3, nom: 'Informatique' },
    { id: 4, nom: 'Marketing' },
    { id: 5, nom: 'Ressources humaines' },
];

export default _departements;
```

Pour la **vue**, nous retrouvons un système similaire à l'exercice du cours 6, soit un tableau HTML dont le corps est composé d'un employé par rangée, où sont affichées toutes les informations de celui-ci. On retrouve également un champ de recherche permettant à l'utilisateur de filtrer les employés par matricule. Nous suggérons

Pour le **contrôleur**, il s'agit de programmer la logique en faisant le lien entre le modèle et la vue de l'application.

Vous avez été sélectionné(e) pour effectuer la programmation de cette application! Suivez les étapes ci-dessous qui expliquent en détails le travail à faire.

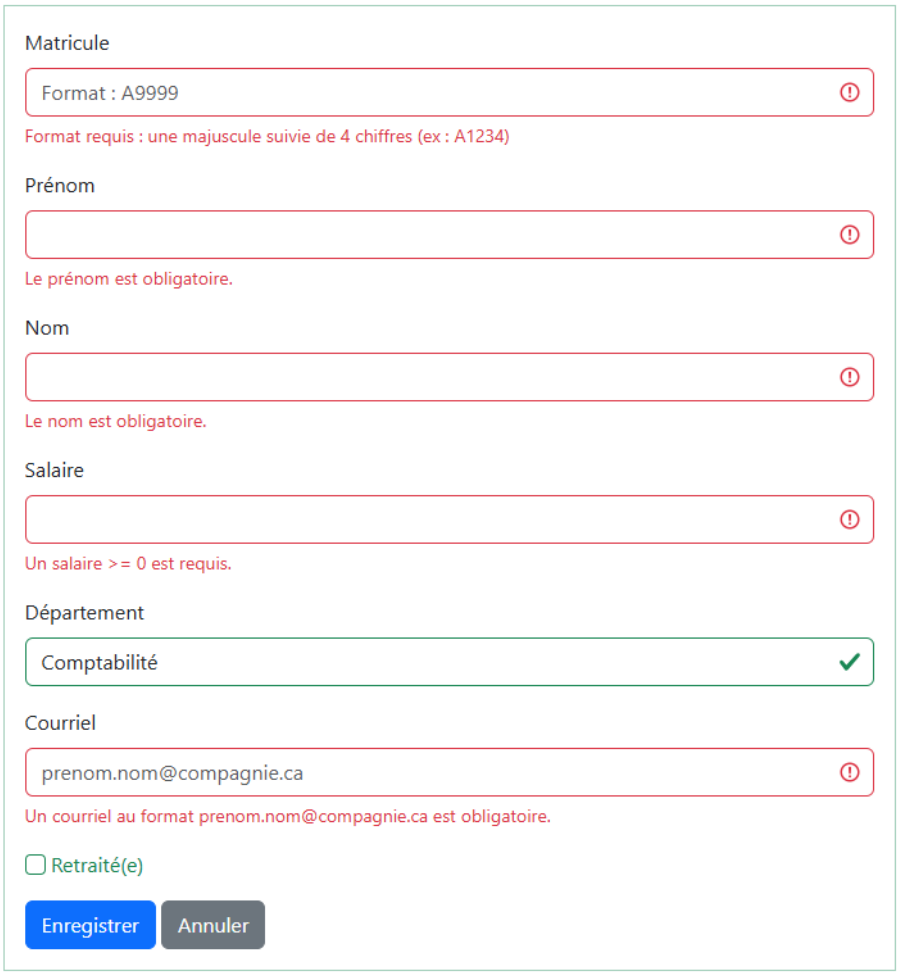
Étape 1 – Compléter le formulaire HTML

Le formulaire a été partiellement conçu mais il est incomplet. Vous devez y ajouter l'attribut *novalidate* afin d'utiliser les classes de validation Bootstrap (version 5.3) et d'ajouter des validations personnalisées.

À partir du tableau suivant, vous devez ajouter les attributs manquants aux divers éléments du formulaire afin de vous conformer aux exigences techniques qui y sont décrites.

Description	**Nom (*name)***	**Type**	**Obligatoire**	**Format exigé**
Matricule	matricule	Texte	Oui	Une lettre majuscule et 4 chiffres (5 caractères)
Prénom	prenom	Texte	Oui	Non vide
Prénom	nom	Texte	Oui	Non vide
Salaire	salaire	Nombre	Oui	Entier, supérieur ou égal à 0
Département	departement	Liste déroulante	Oui	Un des département
Courriel	courriel	Courriel	Oui	Format 'prenom.nom@compagnie.ca'
Retraité(e)	retraite	Case à cocher	Non	
Date de retraite	dateRetraite	Date	Oui (si retraité)	Date antérieure à la date courante

Vous devez également saisir des messages appropriés pour les erreurs de validation dans les éléments <div.invalid-feedback> permettant de fournir une rétroaction à l'utilisateur. Voir les images suivantes pour des exemples de messages pertinents :



* Pour les valeurs possibles des départements, vous devez ajouter dynamiquement les éléments <option> dans l'élément <select> présent dans la page dans le formulaire d'ajout/modification d'employé. Les *valeurs* des options seront un entier (1 à 5) et les *valeurs* sont les noms des départements. Ces informations sont contenues dans un tableau d'objets nommé *_departements* présent dans le fichier *Departements.js*.

Étape 2 – Programmer l'application selon le modèle MVC

Pour ce faire, nous vous suggérons fortement de vous baser sur la solution de l'exercice du cours 6 (gestion des produits) et de l'adapter pour les besoins de ce travail. Il est à noter que si vous n'appliquez pas le modèle MVC dans votre application, une pénalité allant jusqu'à 20% (-2,5 points sur 12,5) vous sera imposée.

Modèle
Vous devez concevoir le modèle des données de l'application. Vous disposez déjà du fichier *Departements.js*, il s'agit de l'importer au bon endroit. Suggestion : faire une classe pour les objets (*Employe.js*) et une autre classe pour faire le lien avec le contrôleur (*ModeleEmployes.js*) en y incluant des méthodes pour :

- Un constructeur avec un attribut qui est un tableau d'objets de la classe *Employe*;
- Retourner le tableau des employés (complet);
- Retourner le tableau des employés (filtré par matricule);
- Retourner le tableau des départements;
- Ajouter ou modifier un employé au tableau des employés.

Vue
Vous devez concevoir la vue de l'application, qui sera responsable de mettre à jour l'affichage de l'application en fonction des données en provenance du contrôleur. Suggestion : faire une classe (*Vue.js*) et y inclure des méthodes pour :

- Montrer le formulaire d'ajout / modification;
- Cacher (et vider) le formulaire d'ajout / modification;
- Mettre à jour la liste des employés à afficher;
- Ajouter les <options> du <select> pour choisir un département.

Contrôleur
Vous devez inscrire les instructions JavaScript nécessaires au bon fonctionnement de l'application dans le fichier *Controleur/Controleur.js*.

Voici les tâches à réaliser à cette étape :

- Réaliser les importations nécessaires : le modèle (classe *Vue*) et la vue (classe *Employes*);
- Définir des variables utiles pour l'application : formulaire, champs, conteneurs, boutons, etc. À vous de définir ce qui vous plaît;
- Déclarer un écouteur d'événement *DOMContentLoaded* sur l'objet *document* dans lequel vous définissez
    - Les écouteurs d'événements suivants :
        - Au clic du bouton 'Ajouter un employé' (vert)
          Il s'agit de faire appel à la bonne méthode de la classe *Vue* pour afficher le formulaire.
        - Au clic d'un bouton 'Modifier' (jaune)
          Il s'agit de faire appel à la bonne méthode de la classe *Vue* pour afficher le formulaire avec les infos de l'employé déjà saisies dans le formulaire.
        - Au *reset* du formulaire d'ajout/modification d'employé

Il s'agit de faire appel à la bonne méthode de la classe *Vue* pour fermer le formulaire.

- <u>À l'envoi (*submit*) du formulaire d'ajout de client (bouton 'Enregistrer')</u>
  Voici l'aperçu des opérations à faire ici
  - Prévenir le comportement par défaut;
  - Vérifier que le formulaire est valide (via *.checkValidity()*) et si ce n'est pas le cas, interrompre l'exécution car l'utilisateur doit apporter des correctifs aux données;
  - Ajouter la classe Bootstrap *was-validated* au formulaire;
  - Effectuer nos validations personnalisées
    - S'assurer que le matricule respecte le format 1 majuscule + 4 chiffres (pour 5 caractères);
    - S'assurer que le prénom n'est pas vide;
    - S'assurer que le nom n'est pas vide;
    - S'assurer que le salaire est >= 0;
    - S'assurer que la ville n'est pas vide;
    - S'assurer que le courriel respecte le format *prenom.nom@compagnie.ca*;
  - Si toutes les validations sont rencontrées :
    - Mettre à jour la vue avec le nouveau client créé;
    - Fermer le formulaire d'ajout / modification.
- <u>Au changement (événement *input*) du champ pour filtrer par province</u>
  Il s'agit de composer une méthode de la classe *Vue* avec une de la classe *Clients* afin de mettre à jour l'affichage de la liste des clients avec une liste filtrée.
- <u>Au changement de la case à cocher 'Retraité(e)' du formulaire</u>
  Si la case est cochée, il faut montrer le champ *dateRetraite* et lui ajouter l'attribut *required* et si la case est décochée, il faut cacher le champ *dateRetraite* et lui retirer l'attribut *required*.

- Les instructions suivantes :
  - Remplir dynamiquement le <select> des départements via un appel à une méthode de la *Vue*.
  - Faire un appel approprié à la méthode de *Vue* qui permet de mettre à jour la liste au chargement initial de la page.
  - Ne pas oublier de gérer les écouteurs d'événements des boutons de modification (jaunes) lorsque la vue est mise à jour!

<u>Remise</u>
Le travail est à remettre le **vendredi 21 mars 2025** en **fin de journée** (23h55m00s).

Bon travail!