

# Test technique Front-End et Back-End

# pour Valentin SALOME

Début du test : 2021-06-14 Date de rendu cible : 2021-06-21

#### **Contact PALO IT**

Hammed Ramdani hramdani@palo-it.com 06 80 22 20 70



- Il s'agit de concevoir une application interactive et ergonomique
- Le langage de programmation suggéré est le JS pour le front, et JS / NodeJS pour le Back. Mais si tu préfères un autre langage tu es libre de le choisir et d'expliquer ce choix
- Les exercices sont à réaliser dans l'ordre de ton choix
- Des implémentations partielles des solutions sont acceptées et ne sont pas éliminatoires, le but étant de discuter des techniques et solutions utilisées ou envisagées pour apprécier la culture de programmation ainsi que la capacité à explorer des solutions différentes et à apprendre
- Pour la partie data, tu peux utiliser une base de données SQL (MySQL, PostgreSQL, etc.) ou NoSQL (CouchBase, MongoDB, Redis, etc.) et expliquer ce choix.
- Publie ton travail sur ton compte Github ou Gitlab pour le partager avec nous. Le code doit être accompagné d'un minimum de documentation, par exemple au format Mark Down
- A toi de jouer :)



# Exercice 1: Jouer avec des Tags:)

- 1. Coder une page web comprenant 2 zones
- 2. Zone 1 : contient un champs (mono-ligne) et un bouton "OK"
- 3. Zone 2 : contient de 0 à N "tag", selon le texte saisi dans le champ
- 4. S'il contient un ou plusieurs noms de villes alors il faut créer un tag pour chaque nom de ville et ignorer les autres mots
- 5. Chaque tag contient le nom de la ville et un picto "X", qui sert à supprimer le tag
- 6. Tu peux utiliser Bootstrap pour aller vite avec les élément CSS
- 7. Tu peux utiliser Vue.JS ou React.JS pour animer la logique de tes pages

Zone 1	lyon toulouse	OK
	lyon x toulousex	



# Exemple de déroulé :

Etape 0 : Saisie	arbre lyon toulouse maison OK
Etape 1 : Validation => maj des tags et du texte	lyon toulouse OK lyon (x) toulouse(x)
Etape 2 : Suppression	lyon toulouse OK  lyon x toulouse x
Etape 3 : MAJ des tags et du texte	toulouse OK toulouse X

Villes à prendre en compte : lyon, paris, marseille, toulouse, lille, bordeau



#### Exercice 2: A nous les données et les WS REST

- 1. Créer une base de données SQL ou NoSQL.
- 2. Créer la structure de données suivante :

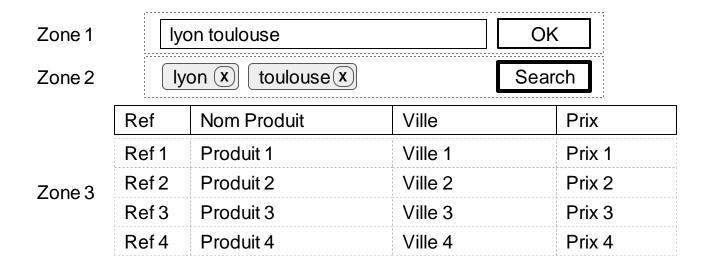
Ref	Nom Produit Ville		Prix
Ref 1	Produit 1	Ville 1	Prix 1
Ref 2	Produit 2	Ville 2	Prix 2
Ref 3	Produit 3	Ville 3	Prix 3
Ref 4	Produit 4	Ville 4	Prix 4

- 3. Créer des WebServices REST pour
  - créer un produit (données à passer : prodRef, prodName, prodCity, prodPrice)
  - modifier un produit (données à passer : prodRef, prodName, prodCity, prodPrice)
  - consulter un produit (données à passer : prodRef)
  - supprimer un produit (données à passer : prodRef)
  - lister les produits (données à passer : rien)
  - lister les produits par ville (données à passer : prodCity = ville1 ou prodCity = ville1, ville2, ...)



# **Exercice 3:** Cherchons les produits

- 1. Ajouter un bouton Search.
- 2. Ajouter une zone liste de produits (zone 3).
- 3. Appeler un WebService REST avec les "tags" comme filtres pour rechercher les produits par ville.





#### Exercice 4: A nous le CRUD

- 1. Enrichir encore la page web initiale
- 2. Ajouter un bouton Add, et gérer la fonction de création de produit grâce à un WebService
- Ajouter le bouton "Modifier" et gérer la fonction d'afficher le formulaire avec les données du produit sélectionné et modifier la donnée après validation
- 4. Ajouter le bouton "Supprimer" et gérer la fonction de suppression du produit sélectionné

Zone 1	lyon toulouse		OK		
Zone 2	lyon x toulouse x			Search	
Zone 3	Ref	Nom Produit	Ville	Prix	
	Ref 1	Produit 1	Ville 1	Prix 1	
	Ref 2	Produit 2	Ville 2	Prix 2	<i>i</i>
	Ref 3	Produit 3	Ville 3	Prix 3	
	Ref 4	Produit 4	Ville 4	Prix 4	

Add



# Exercice 5: Testons tout ça

- 1. Implémenter un ou plusieurs tests unitaires (tu peux t'aider d'une librairie de ton choix).
  - a. Documenter ces test (descriptions, données entrantes, données sortantes, résultats attendus)
  - b. Une commande dans le package.json doit permettre d'exécuter ces tests
  - c. Le retour doit correspondre à la documentation (résultat attendu)
  - d. Dans la mesure du possible, essayer de réaliser au moins un test frontend et un test backend.



# **Exercice 6 :** Oser l'interprétation !

- 1. Pour simplifier la vie de l'utilisateur et rendre l'application "cool":), le Product Owner te demande d'implémenter un interpréteur d'expressions arithmétiques pour permettre à l'utilisateur, lors de la saisie du prix, de tapper des expressions libres du type :
  - a. 10,50+2,30
  - b. 25,60-13,1
  - c. 5\*2
  - d. 100/3
  - e. 2\*5+10,65
  - f. (10+5+2,75-3)\*3

L'objectif est de calculer le résultat - s'il l'expression est bien formulée - et de l'afficher