Intra formatif (Ancien sommatif)

# 420-4W6

## Consignes

* Vous avez 1h50 pour compléter l’examen.
* Toute documentation est permise, mais attention au plagiat.
* Si vous n’avez pas les informations nécessaires pour faire une question, simplement « hardcoder » les données à afficher pour peut-être obtenir des points partiels.
* Si vous n’arrivez pas à compléter une fonctionnalité, faites-en le plus possible, quitte à mettre du code en commentaire pour montrer vos démarches et tentatives.

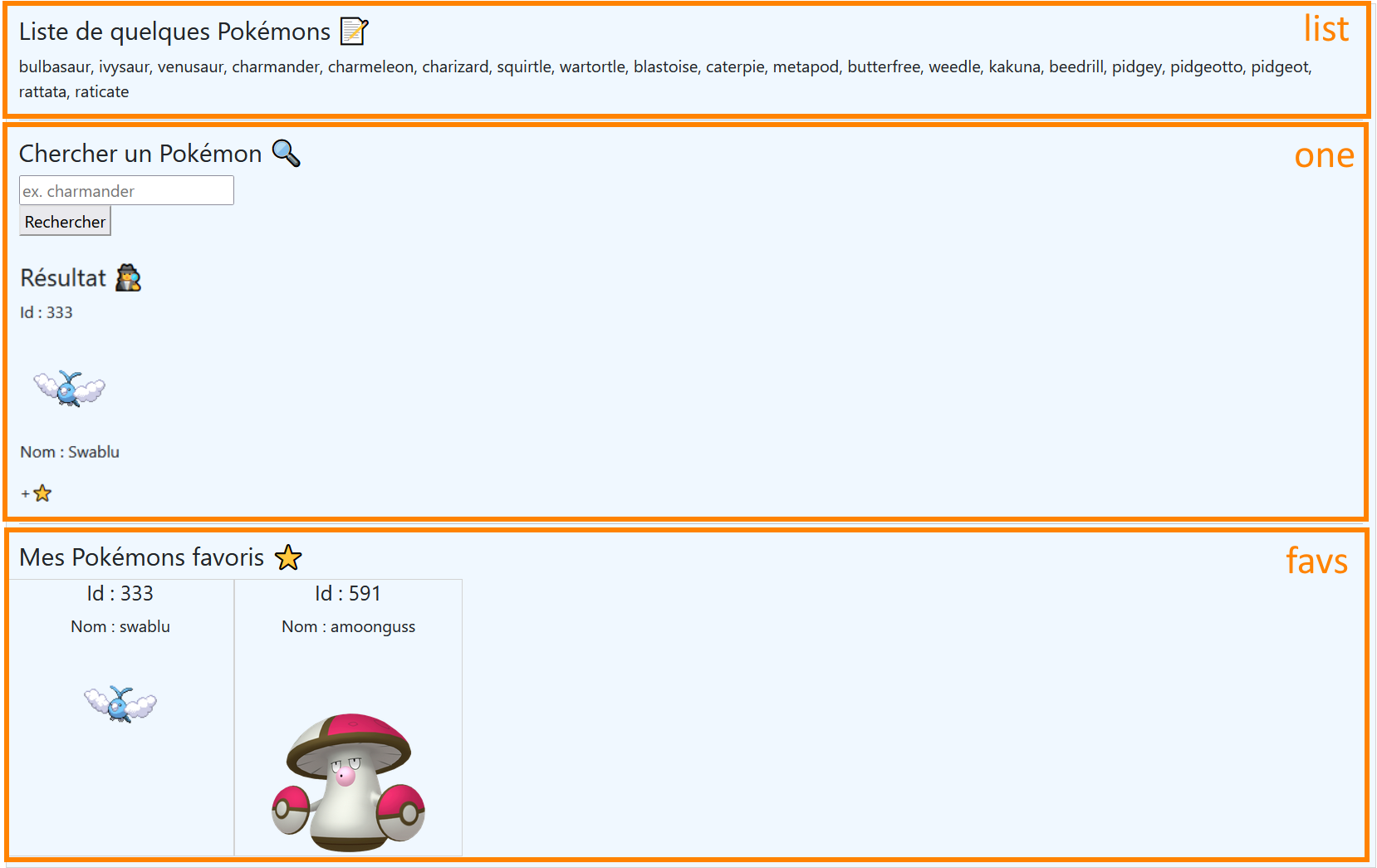
## Note

* Vous devrez inclure les modules nécessaires dans app.module.ts
* Il n’y a aucun package npm à installer. Si un package est nécessaire, il est déjà installé.
* Le CSS n’est pas évalué.

# Projet de départ

Dans le projet de départ, il y a déjà 4 composants :

* **app** : Squelette de la page (header, nav, footer et intégration des 3 autres composants)
* **list** : Affiche une liste de quelques noms de Pokémons.
* **one** : Affiche les informations d’un Pokémon précis qui a été recherché par l’utilisateur.
* **favs** : Affiche les Pokémons favoris de l’utilisateur.



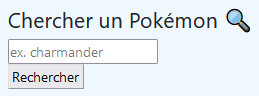
# Question 1 (5 points)

Actuellement, trois composants sont affichés dans le composant **app**. À l’aide du routage, affichez seulement un composant à la fois en ajoutant des règles de routage et en modifiant le composant **app**.

Notez que par défaut, on souhaite afficher le composant **list**. N’oubliez pas de rendre le menu de navigation fonctionnel.

# Question 2 (8 points)

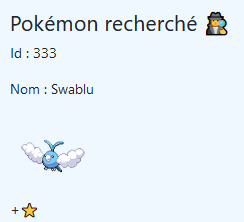
Le bouton « Rechercher » dans le composant **one** doit lancer une requête lorsqu’on clique dessus en utilisant le nom de Pokémon fourni par l’utilisateur dans l’input.



Dans le service **pokeapi**, qui est déjà créé, ajoutez une nouvelle requête permettant d’obtenir les informations d’un Pokémon spécifique :

https://pokeapi.co/api/v2/pokemon/**nom\_d\_un\_pokémon\_en\_minuscules**

Imprimez les données JSON obtenues dans la console. Lorsque l’utilisateur appuie sur « Rechercher » les informations du Pokémon doivent être affichées dans la page. (id, nom et image)



Remarques :

* Il y a déjà l’affichage du Pokémon Swablu à titre d’exemple, mais il doit être remplacé par votre affichage.
* Il existe déjà un *Model* pour les Pokémons dans le code. (Dossier « models ») Il faut utiliser ce modèle.
* Vous aurez le choix entre beaucoup d’URLs d’images : choisissez celle nommée « front\_default » dans la section « sprites ».

# Question 3 (3 points)

Dans le composant **favs**, l’affichage de deux Pokémons *hardcodés* est déjà présent et sert d’exemple dans la page. Remplacez ces deux Pokémons en affichant les trois Pokémons dans la variable « pkmns » du composant. (Il ne faut pas *hardcoder* les trois affichages, il faut se servir de la variable « pkmns » et faire une boucle pour afficher les trois)



# Question 4 (4 points)

Lorsqu’on clique sur « +⭐ » dans le composant **one**, il faut ajouter le Pokémon recherché aux favoris. Dans le composant **favs**, remplacez le tableau avec trois Pokémons *hardcodés* par un nouveau tableau qui contient seulement les Pokémons favoris ajoutés par l’utilisateur à l’aide de l’option « +⭐ ». Notez que ces favoris doivent être persistants même si on change de page ou ferme l’application, donc il faudra utiliser le **stockage local** pour sauvegarder et charger notre liste de favoris.

# Grille de correction

|  |  |
| --- | --- |
| **Section** | **Points** |
| Question 1 | |
| Règles de routage | 3 |
| Modification du menu de navigation et du composant app | 2 |
| Question 2 | |
| Lecture de l’input et lancement de la requête via un clic sur un bouton | 2 |
| Requête et usage d’un Model fourni | 4 |
| Affichage des données | 2 |
| Question 3 | |
| Affichage des données | 3 |
| Question 4 | |
| Ajout aux favoris et sauvegarde | 2 |
| Chargement et affichage des favoris | 2 |
| **Total** | **20** |