




Projet - Gestionnaire de Restaurant



Objectif :

Vous devez créer une application Angular qui permet aux restaurateurs de gérer leurs menus et plats. L'application offrira une interface intuitive pour ajouter, visualiser et gérer des menus et les plats qui les composent.

Travail à rendre :

-  le **build** de votre application
-  le **code source complet** de l'application
-  (un *petit dossier de conception*)

Consignes de livraison :

Votre projet sera livré sous forme d'un **fichier zip** qui devra contenir :

- tout le projet Angular (fichiers de configuration + **src** + **dist**)
 - vous devez supprimer les répertoires **.angular** et **node_modules**
- *mais il faut bien garder tout le reste dans votre zip*
- le mini dossier de conception
- un build avec le projet dans sa dernière version

Attention

Seul votre **build** sera évalué **pour le côté fonctionnel** de l'application, **il doit donc être à jour.**

Le dossier de conception contiendra :

- la liste des fonctionnalités minimales développées
- la liste des fonctionnalités minimales non développées (si il y en a)
- la liste des fonctionnalités bonus développées (si il y en a)
- une présentation rapide de vos classes (composants, services et autres classes)
 - pourquoi cette classe existe, quand est-elle utilisée et que fait-elle ?
- une présentation rapide de l'ergonomie de l'application
 - liste des routes et passage de l'une à l'autre

Date limite de rendu :

- **sans pénalité** : avant le **Dimanche 16 février à 23h55**
 - **- 2 points de pénalité** : avant le Dimanche 22 février 23h55
-

Consignes techniques

Pour démarrer rapidement vos développements, vous pouvez partir du projet angular disponible sur Moodle : `prj-restaurant-2025`

- c'est un projet Angular v18
- il est configuré avec Bootstrap, Bootswatch et FontAwesome (comme au démarrage du TP-todo)

Votre application doit **impérativement utiliser l'API** qui vous est donnée dans le projet `bdd-restaurant` sur Moodle.

Vous ne pouvez pas modifier cette API, car elle sera utilisée pour corriger tous vos projets. Si vous ajoutez des fonctionnalités nécessitant de la persistance dans les options bonus de votre application, vous devrez donc enregistrer ces informations en dehors de l'API (dans le local storage par exemple).

Fonctionnalités minimales demandées :

- Voir la liste des menus
- Voir le détail d'un menu avec la liste de ses plats
- Pouvoir ajouter un plat dans un menu **
- Pouvoir modifier un plat dans un menu **
- Pouvoir supprimer un plat d'un menu
- Pouvoir créer un nouveau menu
- Pouvoir modifier un menu
- Pouvoir supprimer un menu (en supprimant bien entendu ses plats en même temps)

** **NB:** quand on ajoute ou modifie un plat dans un menu, on peut lui donner ou lui modifier librement toutes ses caractéristiques (nom et calories).

✳ Fonctionnalités bonus possibles :

- Filtrer les menus actifs ou non
- Enrichir la page de détail d'un menu en affichant le nombre total de calories du menu (somme des calories des plats)
- Utiliser la table `types_plats` pour aider à donner le nom d'un plat (mais il faut pouvoir modifier ce nom dans le plat)
 - *Il existe de nombreuses possibilités pour obtenir ce genre de fonctionnalités... Si vous n'avez pas d'idée de comment faire regardez la documentation de la balise HTML `<datalist>` sur developer.mozilla.org*
- Marquer les menus du jour comme "servis" ou "non servis"
 - Un bouton pour tout remettre à "non servi" serait le bienvenu
 - Dans un menu, pouvoir tout marquer d'un coup pourrait également être utile

Vous pouvez également proposer d'autres fonctionnalités qui seront évaluées et notées si elles sont présentées dans votre dossier de conception.

L'API **bdd-restaurant**

⚠ **Prenez le temps d'analyser et de comprendre l'API avant de commencer votre projet**, pour éviter de partir dans une mauvaise direction (**l'utilisation de cette API est obligatoire**).

Gestion des menus :

- (`http://localhost:3000/menus`) :
 - en **GET**, donne la liste des menus
 - en **POST**, ajoute un menu
- (`http://localhost:3000/menus/1`) :
 - en **GET**, donne les infos sur le menu ayant l'ID 1
 - en **PUT**, modifie le menu ayant l'ID 1
 - en **DELETE**, supprime le menu ayant l'ID 1 **mais pas ses plats**

Gestion des plats :

- (`http://localhost:3000/menus/2/plats`) :
 - en **GET**, donne la liste des plats du menu ayant l'ID 2
 - **Attention le format de cette URL est différent des autres**
- (`http://localhost:3000/plats`) :
 - en **POST**, ajoute un plat
 - **Attention** il faut bien préciser l'ID de son menu dans vos données
 - Exemple (si on l'ajoute au menu 2) :

```
{
  "menuId": 2,
  "nom": "Salade César",
  "calories": 300
}
```

- (`http://localhost:3000/plats/18`) :
 - en **GET**, donne les infos sur le plat ayant l'ID 18
 - en **DELETE**, supprime le plat ayant l'ID 18
 - en **PUT**, modifie le plat ayant l'ID 18
 - **Attention**, l'URL ne précise pas l'ID du menu de ce plat, il doit toujours être bien précisé dans vos données comme lors de la création du plat.

Gestion des types de plats :

- (`http://localhost:3000/types_plats`) :
 - en **GET**, donne la liste des types de plats
 - *il n'y a pas d'autres actions à faire sur cette table, elle permet seulement de manière optionnelle d'aider à saisir le nom des plats*

Des idées de stratégies

Il est conseillé de travailler par étapes :

1. **Comprendre la structure de données de l'API**
2. **Faire une maquette (papier) de votre application**
 - Chaque page (nom + contenu)
 - Les liens entre les pages
3. **Définir la liste de tous les composants nécessaires**
 - Et choisir une politique de nommage pour bien les identifier
4. **Définir la liste des classes métier**
5. **Définir la liste des services**

 **À ce stade, vous pouvez commencer à coder !**

- Créez toutes les classes
- Créez tous les services
- Créez tous les composants
- Développez les classes
- Développez les services
- Définissez toutes les routes
- Définissez un ordre pertinent pour développer les composants
- Puis développez-les **un par un**

.

.

.

.

.

.



Comment on est noté ?

La note sur 20 est basée sur la réalisation du cahier des charges ci-dessus.

La grille de notation :

- Fonctionnalités minimales :	
- Voir la liste des menus	: 2 points
- Voir le détail d'un menu	: 2 points
- Ajout d'un plat dans un menu	: 2 points
- Modification et suppression d'un plat	: 2 points
- Création d'un menu	: 2 points
- Modification d'un menu	: 1 point
- Suppression d'un menu et de ses plats	: 1 point
- Fonctionnalités bonus	: 4 points
- Document de conception (2 points)	
- Qualité du code et ergonomie	: 4 points

Le projet est donc :

- noté sur 18 sans aucune fonctionnalité bonus
- noté sur 22 au maximum avec des fonctionnalités bonus
- mais la note ne dépassera pas 20 sur scodoc 🤔
- NB: **avoir développé une fonctionnalité ne garantit pas d'avoir tous les points de cette fonctionnalité** (il faut pour cela qu'elle soit fonctionnelle et bien développée)



noté sur 18 sans bonus, peut aller jusqu'à 22 (mais limité à 20 sur Pronote).