# Examen Angular

Vous souhaitez vous lancer dans le marketing du jeu-vidéo, pour ceci, vous avez accès à une API de données sur le domaine.

## Le sujet principal

Vous aurez à réaliser plusieurs pages pour votre application (total de 4) :

- Un component permettant d'afficher le **store**.
  Le store permet d'afficher la liste de tous les jeux disponibles.
  Vous devrez gérer la pagination, afin de ne pas charger tous les jeux d'un coup et faire attendre l'utilisateur.
  Utilisez des « **cards** » personnalisées pour les différents jeux, elle doit être un récapitulatif du jeu (avec ou sans **@Input**, libre à vous).

- Un component récapitulatif de **tous les comptes** existant dans l'application, vous afficherez tous les attributs de chaque compte.

- Un component de **création de compte**, qui reprends juste les champs : **email** et **nom de compte.** Lorsque l'on créé un utilisateur, sa wallet (portefeuille, est toujours initialisé 0). Un simple console log me suffit pour valider la création d'un utilisateur.

- Lorsque l'on arrive sur l'application, on doit voir le store en première page.

- Vous ajouterez un bouton permettant d'aller sur la vue des **tous les comptes**, de **création de compte** et pour retourner à l'accueil (**store**).

## Le sujet supplémentaire

Une fois le sujet principal terminée, vous pouvez améliorer votre application pour avoir des points bonus et augmenter votre note. **Cette partie ne compte pas dans la notation « standard » et est donc facultative, mais rien ne vous empêche d'en faire avant le reste.**

- Component de détail d'un jeu **(nouveau component à créer)** :
  - le but de ce component est d'afficher le détail d'un jeu, avec son image principale, et le reste de ses informations.
  - Il doit inclure les commentaires relatifs à ce jeu

- Component de création d'un compte :
  - Ajouter l'utilisateur créé en base de données

- Component de détail d'un compte (**nouveau component à créer**) :
  - Lorsque l'on clique sur un utilisateur depuis le **component de tous les comptes**, on arrive sur ce component.

- Il doit afficher toutes les informations d'un utilisateur et les différents jeux de sa bibliothèque.

## Rendre le projet

Vous déposerez votre projet sur un repo git perso, puis vous me donnerez le lien via Discord.

Vous penserez à m'ajouter sur votre repository, afin que je puisse le visualiser et potentiellement push des trucs dessus. (tourret.kevin@gmail.com)

Vous avez accès à tous les cours, support de cours disponibles, aux projets de cours et à internet. **Cependant, vous devez le réaliser individuellement.**

## Notation

| Critère | Point |
|---|---|
| « ng serve » sans erreur terminal | 1 |
| « ng serve » sans erreur navigateur | 1 |
| Création de component | 1 |
| Création de services pour l'API | 5 |
| Formulaire (code ou template) | 2 |
| Validation formulaire | 2 |
| Utilisation d'interface et classe | 3 |
| Présence de lien entre component | 2 |
| Présence d'Input | 2 |
| Bonne organisation de l'application | 2 |
| Interface graphique ergonomique (facile à utiliser) | 2 |
| Bonus graphique (beau) | 2 |
| Sujet principal | 7 |
| Sujet supplémentaire | 10 |
| Bonus en tout genre | Variable entre 1 et 3, sujet mené à bien de A à Z, votre petite touche personnelle dans l'application et le code… (c'est plus agréable pour corriger 😉)<br>**[ces points ne comptent pas dans la note maximale]** |
| Malus de correction | Valeur négative, variable selon les erreurs (pas de limite, si je dois modifier du code pour faire fonctionner le projet, présence de « any » etc) |
| **Total** | **42 (hors bonus en tout genre)** |

## Documentations

Explications des objets de l'API (**les champs sont tous obligatoire sauf contre-indication**) :

- **comments** : table des commentaires des jeux, un commentaire est relatif à un jeu et un utilisateur peut mettre un commentaire sur un jeu (mais pas forcément sur un jeu qu'il possède !)
  - **account_id** : l'id du compte ayant écrit le commentaire
  - **game_id** : l'id du jeu relatif au commentaire
  - **create_at** : date de publication du commentaire
  - **content** : contenu du commentaire
  - **up_votes** : nombre de votes positif reçus par le commentaire
  - **down_votes** : nombre de votes négatif reçus par le commentaire

- **accounts** : table des comptes des utilisateur
  - **name** : nom du compte (**unique**)
  - **email** : email de création du compte (**unique**)
  - **nickname** : pseudonyme du joueur, affiché publiquement
  - **wallet** : l'argent actuellement présent sur son compte

- **games** : table des jeux
  - **name** : nom du jeu
  - **published_at** : date de sortie du jeu
  - **price** : prix de ventes du jeu
  - **thumbnail_cover** : image principale du jeu (nullable !)
  - **thumbnail_logo** : image logo du jeu (nullable !)
  -

- **libraries** : table représentant une entrée de la bibliothèque des joueurs
  - **game_id** : nom du jeu présent dans la bibliothèque
  - **account_id** : compte à qui appartient à bibliothèque
  - **installed** : le jeu est-il installé ?
  - **game_time** : temps de jeu en **secondes** de l'utilisateur sur le jeu
  - **last_used** : date de dernière utilisation du jeu par l'utilisateur

- **languages** : table représentant les langues dans lesquelles peuvent être disponible les jeux
  - **name** : nom de la langue

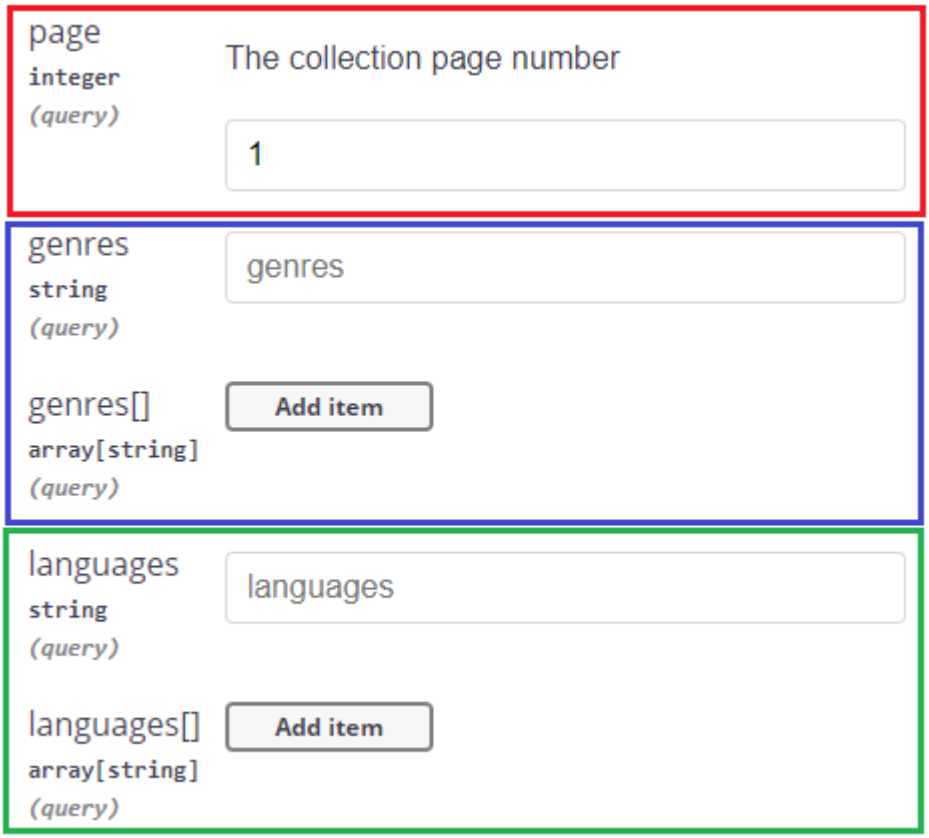
- **genres** : table représentant les genres possibles des jeux
  - **name** : nom du genre

Lien vers l'API : https://steam-ish.test-02.drosalys.net/api

Aide pour utiliser l'API :

Si par exemple vous souhaitez récupérer tous les « **games** » de l'application :

- https://steam-ish.test-02.drosalys.net/api/games
  Pour récupérer les datas du Json, il vaut les ressortir depuis le champ « **hydra:member** »
- Vous noterez que certains liens ont des filtres possibles, par exemple game :




Une pagination est intégrée à API platform (rouge), vous pouvez la récupérer via cette URL : https://steam-ish.test-02.drosalys.net/api/games?page=1

Du coup pour les genres et langages (bleu et vert), il faut faire : https://steam-ish.test-02.drosalys.net/api/games?genres=/api/genres/1

Il faut reprendre « /api/genres/**ID** » et pour les langues ? Il faut mettre « **languages** » à la place de « **genres** ». Etudiez bien la documentation de l'API platform…

## Aide pour faire l'examen :

Faites-le, ou ne le faites pas, mais je vous recommande d'utiliser une interface **IApiPlatform, de cette forme :**

```
export interface IApiPlatform<T> {
  'hydra:member': Array<T>;
  'hydra:view': {
    'hydra:last' : string;
  }
}
```

Cela devrait vous faciliter la tâche pour récupérer les données de l'API…

Le **<T>** représente le type que vous voulez, c'est-à-dire que si vous voulez faire une fonction qui récupère les langues de l'application, vous aurez quelque chose comme ça :

```
getLanguages(): Observable<IApiPlatform<Languages>>
```

(PS : vous n'avez aucune contrainte pour utiliser, ou non, une bibliothèque de style, genre Bootstrap ou… ce que vous voulez ?!)

Bon courage à vous 😉