Bases de Données et Applications

séances 5-6

objectif : vers une API REST

L'objectif est de construire progressivement une API permettant d'accéder aux données de la base GamePedia au travers d'une interface web de type REST.

Partie 1 : Accéder un 1 jeu

En utilisant le mini-framework slim, construire une route qui doit avoir la forme suivante :

http://www.gamepedia.net/api/games/id et qui doit retourner la représentation JSON du jeu n° ***id.***

Cette représentation est un objet json :

```
{
  "id": 35444,
  "name": "War Thunder",
  "alias": "World of Planes",
  "deck": "War Thunder is a free-to-play air combat game set in the World War II and Korean War eras. Naval and land battles are planned additions.",
  "description": "...",
  "original_release_date": "2013-06-30 00:00:00"
}
```

Par ailleurs, la réponse doit contenir le header "Content-Type" avec une valeur positionnée à "application/json".

Partie 2 : accéder à un 1 collection de jeux

Compléter les routes traitées par votre api pour gérer la route suivante :

http://www.gamepedia.net/api/games

qui retourne une collection de jeux. Dans un premier temps, cette route renvoie simplement la liste des 200 premiers jeux. Chaque jeu est un objet json contenant une représentation simplifiée du jeu.

Le résultat doit avoir cette forme :

```
{ "games" : [
        {
         "id": 1,
         "name": "War Thunder",
         "alias": "World of Planes",
         "deck": "War Thunder is a free-to-play air combat game set in the World War II and Korean War
                  eras. Naval and land battles are planned additions.",
        },
        {
         "id": 2,
         "name": "Desert Strike: Return to the Gulf",
         "alias": "Desert Strike Advance",
         "deck": "A top-down isometric helicopter shoot 'em up originally for the Sega Genesis, which was
                   later ported to a variety of platforms. It is best known for its open-ended mission design and
                  was followed by several sequels.",
        }, ...
       ]
}
```

Partie 3 : pagination

Compléter la route précédente pour inclure de la pagination, permettant de parcourir l'ensemble des jeux :

***http://www.gamepedia.net*/api/games?page=23**

On fixe la taille des pages à 200 jeux.

Une fois que la pagination fonctionne, compléter le json retourné pour une page en y ajoutant un lien vers la page précédente et un lien vers la page suivante :

```
{ "games" : [
          { ... },
          { ... }, ...
         ],
  "links" : {
    "prev" : { "href" : "/api/games?page=22" },
    "next" :  { "href" : "/api/games?page=24" }
}
```

Partie 4 : lien collection → ressource

Compléter la représentation d'une collection de jeux en ajoutant, pour chaque jeu de la collection, l'uri du jeu permettant d'accéder à sa description détaillée. Chaque élément du tableau de ressource aura donc la forme suivante :

```
{
          "game" : {
           "id": 1,
           "name": "War Thunder",
           "alias": "World of Planes",
           "deck": "War Thunder is a free-to-play air combat game set in the World War II and Korean War
                    eras. Naval and land battles are planned additions.",
          },
          "links" : {
            "self" : { "href" : "/api/games/1"}
 }
```

La construction de l'uri du jeu **devra** utiliser le mécanisme de nommage de route de Slim.

Partie 5 : commentaires sur les jeux

Implanter la route suivante :

***http://www.gamepedia.net*/api/games/*2*/comments**

qui retourne la collection de commentaires associés à un jeu.

Pour chaque commentaire retourné dans la collection, la représentation contient l'id, le titre, le texte, la date de création et le nom de l'utilisateur.

Partie 6 : retour sur les jeux

Transformer la représentation d'un jeu retournée par l'uri du type
http://www.gamepedia.net*/api/games/*id

pour qu'elle soit plus complète. Elle doit maintenant être organisée en 2 parties :

- une partie contenant la description du jeu en lui même,

- une partie indiquant des liens qu'il est possible de suivre à partir de ce jeu, par exemple le lien permettant d'obtenir les commentaires du jeu.

Cette représentation aura cette forme :

```
{
  "game" : {
          "id": 35444,
          "name": "War Thunder",
          "alias": "World of Planes",
          "deck": "War Thunder is a free-to-play air combat game set in the World War II and Korean War
          eras. Naval and land battles are planned additions.",
          "description": "...",
          "original_release_date": "2013-06-30 00:00:00"
          },
  "links" : {
          "comments" : { "href" : "/api/games/35444/comments"},
          "characters" : { "href" : "/api/games/35444/characters"} }
}
```

Ajouter ensuite, dans la représentation d'un jeu, la liste des plateformes pour ce jeu. Cette liste est tableau correspondant à une collection de plateforme, construit sur le même principe que le tableau de jeux représentant une collection de jeux. Pour chaque plateforme, on retourne son id, son nom, alias et abbréviation, ainsi que l'url vers sa description détaillée.

Partie 7 : les personnages d'un jeu

Implanter le traitement des requêtes du type **/api/games/*id*/characters**

qui retournent la liste des personnages d'un jeu. La liste doit avoir la forme suivante :

```
{ "characters" : [
          { "character" : {
                "id": 1,
                "name": "Scorpion",
                 },
             "links" : {
                "self" : { "href" : "/api/characters/1"}
                }
          },
          { "character" : {
                "id": 2,
                "name": "Sub-Zero",
                 },
             "links" : {
                "self" : { "href" : "/api/characters/2"}
                }
          }
          { ... }, ...
  ]
}
```

Partie 8 : ajouter des commentaires

On veut permettre l'ajout de commentaires sur un jeu au travers de l'api. Cela consiste à implanter la requête POST pour l'uri **/api/games/id/comments** .
Les données pour créer le commentaire sont transmises dans le body de la requête : email de

l'utilisateur, titre et contenu du commentaire.
Requête : Faire en sorte que les données soient transmises en json.
Réponse : en cas de succès , le code de retour http **doit** être **201**. La réponse doit contenir le header **Location:** avec comme valeur l'uri du nouveau commentaire (/api/comments/*id*).
Le body de la réponse doit contenir une représentation json du nouveau commentaire.