Structure interne de la BDD du serveur

Toutes les clés doivent commencer par une majuscule, et suivre ensuite le CamelCase.

Film

Un film est représenté par l'élement suivant:

```
{
"Actors":
    ["",...],
"Description": "",
"Directors":
    ["",...],
"Duration": 0,
"Genres":
    ["", ...],
"Id": 0,
"Title": "",
"Type": "",
"Year": 0
}
```

- Toutes les clés sont forcément définies sur le type donné.
- Si le type donné ne contient pas de valeur, il est assigné a la valeur nulle correspondant au type (valeur par défaut du type, 0, chaine vide, array vide, etc...)
- L'id commence a 1.
- Pour les types int, toute valeur nulle est considérée comme non-présente. (Ex: Année = 0, alors nous n'avons pas l'information sur l'année)
- Idem pour les chaines avec la chaine vide.

User

Un utilisateur est représenté par l'élement suivant:

```
{
    "Id" : 0,
    "Name" : "",
    "FirstName" : "",
    "Login" : "",
    "Password" : "",
    "Birth" : {Date},
    "Preferences" : {Preferences},
    "History" : {History},
    "Friends" : [0, ...], //Id d'un utilisateur
    "Follow" : [0, ...] //Id d'un utilisateur
}
```

- Id ne peut être nul pour représenter un utilisateur réel.
- Login ne peut être nul pour représenter un utilisateur réel.
- Birth permet de stocker la date de naissance, afin de calculer l'age de l'utilisateur (pour cibler le public).
- Preferences permet de stocker les préférences d'un utilisateur dans une structure dédiée (voir plus bas).
- History permet de stocker l'historique de l'utilisateur dans un objet dédié (voir plus bas).
- Friends correspond a la liste d'amis des utilisateurs, ça permet de lui recommander des films que des amis auraient aimé, et de nourrir l'algo de recommendation si celui ci à encore faim.
- Follow permet à l'utilisateur de définir des gens qu'il suit car le contenu qu'ils regardent les interessent. Cela permet d'utiliser des données d'un autre utilisateur pour alimenter celui ci.

Date

Une date est représenté par l'élément suivant:

```
{
    "Day" : 0,
    "Month" : 0,
    "Year" : 0,
}
```

- Peut être utilisé comme Date.

Preferences

Preferences à pour but de stocker les préférences utilisateurs pour nourir l'algorithme de recommandation.

```
{
    "StaticPreferences" : {StaticPreferences},
    "DynamicPreferences" : {DynamicPreferences}
}
```

StaticPreferences

Cet objet a pour but de stocker toutes les préférences que l'utilisateur a choisi de lui même. Tout ce qu'il a explicitement défini comme Aimer, ne pas Aimer, etc...
Ces préférences ne peuvent être modifiées par l'algorithme de recommandation, elles servent de base dure à un profil utilisateur.

```
{
    "Hated" : {
        "Films" : [0, ...],//FilmId
        "Genres" : ["", ...]
        "KeyWords" : ["", ...]
    },
    "Liked" : {
        "Films" : [0, ...],//FilmId
        "Genres" : ["", ...]
        "KeyWords" : ["", ...]
    },
    "TopLiked" : {
        "Films" : [0, ...],//FilmId
        "Genres" : ["", ...]
        "KeyWords" : ["", ...]
    },
    "TopHated" : {
        "Films" : [0, ...],//FilmId
        "Genres" : ["", ...]
        "KeyWords" : ["", ...]
    }
}
```

- Hated contient des listes de choses non aimées par l'utilisateur, sans classement
- Liked contient des listes de choses aimées par l'utilisateur, sans classement
- TopLiked contient un top des choses les plus aimées par l'utilisateur, classé par importance.
- TopHated contient un top des choses les plus detestées par l'utilisateur, classé par importance.

→ L'utilisateur ne peut pas a la fois aimer et ne pas aimer quelque chose.
→ Les listes TopLiked et Liked sont complémentaires. Ainsi l'utilisateur peut dire: Mon top des films préféré est, et j'aime certains genres, sans forcément tout classer.
→ Idem pour TopHated et Hated.

Toutes ces préférences sont modifiables par l'utilisateur.

DynamicPreferences

Ces préférences sont générées par l'algorithme de recommandation, qui les met a jour en fonction des actions utilisateur.
Mettre une bonne note a un film va lui faire déduire qu'il a aimé ce film par exemple.

```
{
    "Hated" : {
        "Films" : [0, ...],//FilmId
        "Genres" : ["", ...]
        "KeyWords" : ["", ...]
    },
    "Liked" : {
        "Films" : [0, ...],//FilmId
        "Genres" : ["", ...]
        "KeyWords" : ["", ...]
    },
    "TopLiked" : {
        "Films" : [0, ...],//FilmId
        "Genres" : ["", ...]
        "KeyWords" : ["", ...]
    },
    "TopHated" : {
        "Films" : [0, ...],//FilmId
        "Genres" : ["", ...]
        "KeyWords" : ["", ...]
    }
}
```

- Structure identique à StaticPreferences.

History

History à pour but de stocker les actions utilisateurs, et évenements le concernant:

```
{
    "Log" : [{HistoryAction}, ...],
    "Viewed" : [{HistoryViewed}, ...],
    "Searches" : [{HistorySearched}, ...],
    "Recommended" : {HistoryRecomendation},
    "OldRecommended" : [{HistoryRecomendation}, ...],
    "Displayed" : {HistoryDisplayed},
    "Rates" : [{HistoryRate}, ...],
}
```

HistoryAction

HistoryAction à pour but de stocker les actions utilisateurs pour les associer a une date afin de permettre de retracer son historique, pour mieux comprendre son comportement.

```
{
    "Time" : 0,//Timestamp
    "Action" : "",//Voir la liste
    "ActionId" : 0
}
```

- Time permet de stocker un timestamp (valeur renvoyée par la fonction `time(0)` en C).
- Action permet de stocker un type d'action, le but est de la retrouver ensuite ailleurs en fonction de son type et son Id.

Actions:

- "Request" → Requête réseau
- "View" → Consulte une fiche
- "Search" → Execute une recherche
- "Rate" → Attribue une note
- "Login" → Connexion à un compte
- "Logout" → Déconnexion

ActionId correspond a l'Id de l'action enregistré par le serveur.

HistoryViewed

Correspond à la consultation d'une fiche par un utilisateur:

```
{
    "FilmId" : 0,//Correspond a l'ID de l'action et du film
    "Times" : [0, ...],
}
```

- FilmId correspond a l'ID du film vu.
- Times correspond a un tableau de timestamp contenant toutes les fois où l'utilisateur a consulté, ce qui permet d'avoir le nombre et la fréquence.

HistorySearched

Correspond à une recherche lancée par un utilisateur:

```
{
    "SearchId" : 0,//Correspond a ActionId
    "Query" : ["", ...],//Recherche
    "FullText" : "",//Recherche (string)
    "Time" : 0
}
```

- Query contient un tableau de mots clés, ce qui permet de facilement associer les mots entre eux. Tout ce qui n'est pas considéré comme mot clé ne sera pas dans le tableau.
- FullText contient la requête originale.

HistoryRecommendation

Contient les recommendations actuelles pour un utilisateur, permet de sauvegarder une liste de recommandations et peut être utilisée pour générée une nouvelle liste.

```
{
    "Id" : 0,
    "Films" : [0, ...],
    "Time" : 0,
    "Genres" : ["", ...],
    "KeyWords" : ["", ...],
    "Used" : [0, ...]
}
```

- Id contient un Id unique pour cette liste de recommandation.
- Films va contenir les Id des films qui font partit de la liste de recommandation, par ordre d'importance. Il n'y a pas de limite de quantité, c'est l'algorithme d'affichage qui effectuera le tri, et une selection.
- Time contient le timestamp de la date de génération de la liste, pour éviter d'utiliser des listes trop anciennes.
- Genres contient la liste des genres associée a tous les films de la liste, pour aider a l'amélioration de la recommandation. (Chaque genre ne peut être présent qu'une fois). Les genres sont classés par ordre d'importance.
- KeyWords cette liste contient, par ordre d'importance, la liste des mots clés utilisés pour générer cette liste de recommandation (si il y en a), c'est notament inspiré des noms de films, ou des recherches de l'utilisateur.

HistoryDisplayed

Contient une liste des films affichés par l'algorithme d'affichage.
Cela permet de répertorier la position d'affichage des films, et leur nombre d'affichage. (Afin de pouvoir effectuer une meilleure selection)

```
{
    "FilmId" : 0,
    "Displayed" : [{
        "Time" : 0,
        "Position" : 0
    }, ...]
}
```

- Displayed contient la liste des fois affichées, et la position.
- La plus petite position est 1, = meilleure position.

HistoryRates

Contient les notes données par un utilisateur a un film.
L'utilisateur peut mettre une note entre 1 et 5, 0 = pas de note.
Et il peut choisir j'aime ou j'aime pas. (Car contradictoirement, certains utilisateur n'aiment pas un film, mais vont lui mettre une bonne note pour certains points, et inversement)

```
{
    "Id" : 0, //FilmId
    "Rate" : 0, //Note
    "Liked" : 0, //Aimé ou pas.
}
```

- Id correspond a l'id du film
- Rate correspond a la note entre 0 et 5
- Liked correspond a 0 = non choisi, 1 = aimé, 2 = pas aimé

Session

Une session est définie selon le schéma suivant:

```
{
    "SessionId": "",
```

```
    "UserId": 0,
}
```