Architecture logicielle, API et interopérabilité

J1 - Intégration API

Partie 1 - Intégrer une API comme fournisseur de données (FD)

Exercice 1 ⭐

Reprendre notre application Python/Django de todolist : https://github.com/gaeldb/to-do-list
Nous allons transformer cette application en une liste de série Netflix / Amazon Prime à regarder.

Faites un fork nommé my-watch-list.

Remettez le nez là-dedans, relancez le serveur, etc.

Exercice 2 ⭐⭐

Implémenter l'API TMDB dans un outil d'appel API (choisissez Bruno https://www.usebruno.com/ ou le Postman), vous devez créer 4 appels :

- GET 10 séries du genre "Action & Adventure"
- GET 10 séries avec les meilleures notes sur Netflix
- GET 10 séries avec les meilleures notes sur Amazon Prime Video
- GET 10 séries avec les meilleures notes sur sur Apple TV

Tips :

- https://developer.themoviedb.org/docs/getting-started
- Utiliser l'API "Discover TV"

Exercice 3 ⭐

Dans l'application, ajoutez 3 boutons :

- Ajouter 10 séries Netflix à ma watchlist
- Ajouter 10 séries Amazon Prime à ma watchlist

- Ajouter 10 séries Apple TV à ma watchlist

Ces 3 boutons doivent effectuer l'action indiquée dans leur nom respectif : appelez l'API correspondante et créer les entrées dans la watchlist.

Exercice 4 ⭐⭐

Maîtrisez les doublons.
Une même série ne doit pas être créée plusieurs fois dans la watchlist.
Si je clique 2 fois sur Ajouter 10 séries Netflix à ma watchlist : je dois avec 20 séries **différentes**.

Tips : utilisez les ID du fournisseur de service pour éviter les doublons.

Rendu 1

Déposez votre code sur Github, et indiquez votre repository dans le tableau suivant :
Github rendus
Il doit être accessible à l'utilisateur gaeldb en lecteur.

Prenez une vidéo pour illustrer les exercices et déposez-là dans une issue Github nommée "Rendu 1". Si besoin de compresser la vidéo, l'outil Handbrake est très bien.

Partie 2 - Intégrer une API comme fournisseur d'identité (FI)

Exercice 5 ⭐⭐

Pour que tout le monde ne puisse pas voir ma watchlist, nous allons implémenter un système de connexion / déconnexion.

Cette évolution doit prendre en compte :

1. je dois être redirigé sur une page de login si je ne suis pas connecté à l'application
2. je dois pouvoir me déconnecter
3. je dois pouvoir créer un compte
4. je dois voir uniquement ma watchlist (pas celle des autres utilisateurs)

Tips :

- utiliser tout ce que Django fournit déjà : https://docs.djangoproject.com/en/6.0/topics/auth/default/
- utiliser au maximum le décorateur @login_required ou son équivalent mixin LoginRequiredMixin

Exercice 6 ⭐⭐⭐

Maintenant que nous avons une gestion des identités interne, nous allons ajouter un fournisseur d'identité externe : nous allons permettre aux utilisateurs de se connecter avec France Connect.

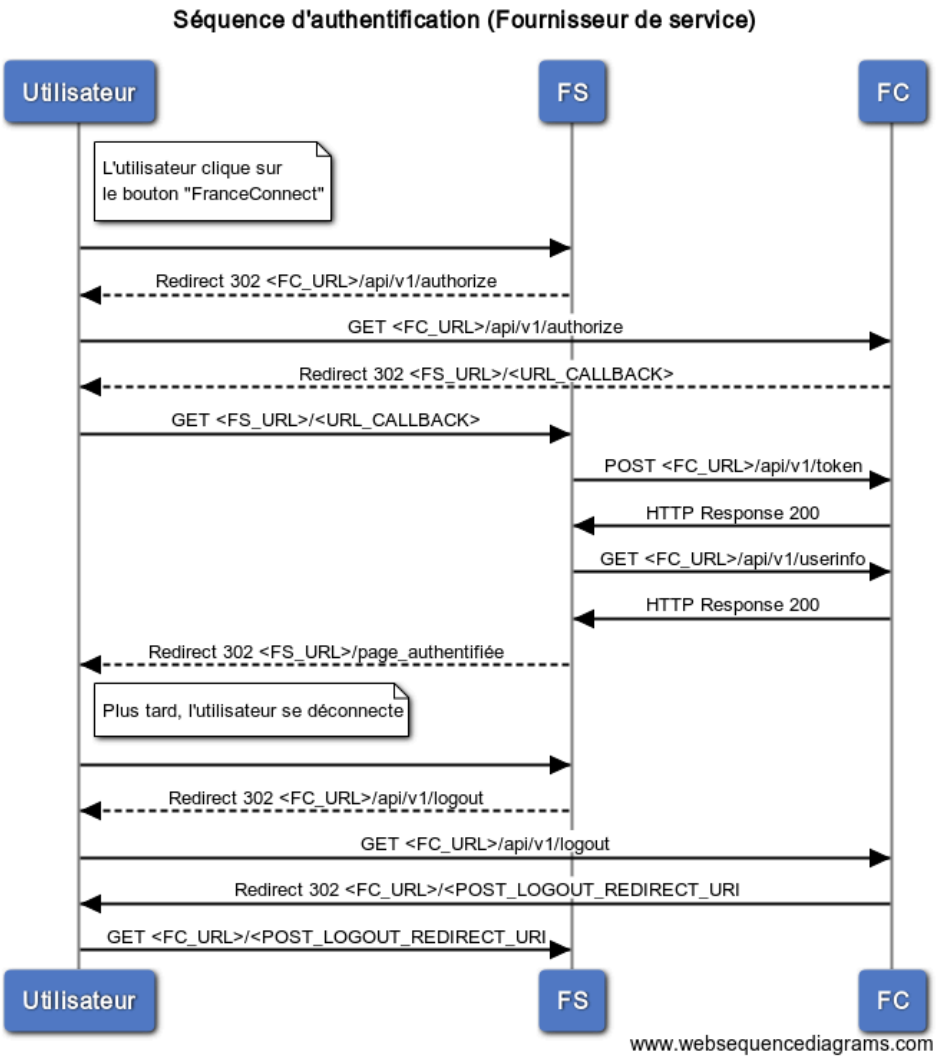
Ajouter un bouton Se connecter avec France Connect sur la page de login.

A l'issue de la séquence d'authentification FC, si l'utilisateur n'est pas encore connu par notre application, son compte est créé automatiquement.

Tips :

- https://partenaires.franceconnect.gouv.fr/fcp/fournisseur-service
- attention, il existe 2 versions de l'API France Connect, v1 et v2. La v1 est plus simple et fonctionne encore.
- les comptes de dev France Connect : https://github.com/france-connect/sources/blob/main/docker/volumes/fcp-low/mocks/idp/databases/citizen/base.csv

- Regardez bien la séquence (ici, l'exemple de la v1) :



Exercice 7 (facultatif bonus) ⭐⭐

Permettre aux utilisateurs de se connecter avec leur compte Google.

Rendu 2

Déposez votre code sur Github, et vérifier que votre repository est dans le tableau suivant :
Github rendus
Il doit être accessible à l'utilisateur gaeldb en lecteur.

Prenez une vidéo pour illustrer les exercices et déposez-là dans une issue Github nommée "Rendu 2". Si besoin de compresser la vidéo, l'outil Handbrake est très bien.

C’est fini, bravo !