

# Objectifs

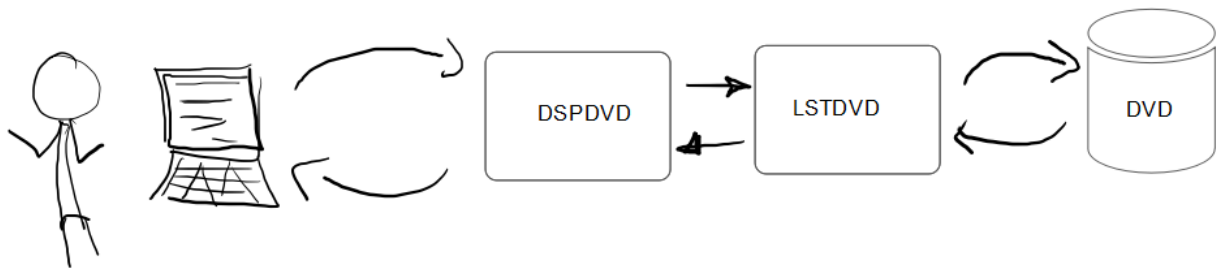
- Consommer un service REST en RPG avec SQL et JSON.

## Énoncés

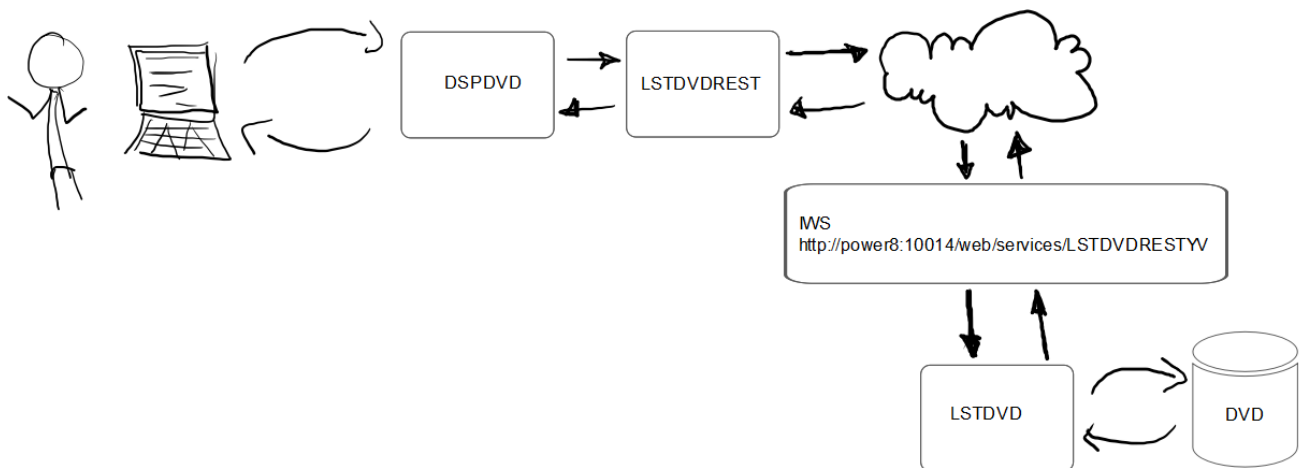
Nous allons consommer le service REST du programme de service LSTDVDREST\*\* que vous avez déployé au TP precedent.

Si vous n'avez pas eu le temps de le déployer vous pouvez utiliser le web service REST LSTDVDRESTYV (ou celui de votre voisin) de WSERVICE2.

Nous allons transformer notre process écrit précédemment pour afficher la liste des DVDS dans le programme DSPDVD.



devient



Cela n'a bien sur aucune réalité professionnelle et cette complication est réalisée à titre d'exercice.

- Liste des DVDs

Nous souhaitons utiliser le WEB service LSTDVDREST\*\* pour réaliser un programme (LSTDVDREST) pour récupérer la liste des 10 premiers DVDs. Pour ce faire :

- nous allons utiliser la structure du programme LSTDVD vu précédemment.
- la méthodologie vue en cours (cf rappel ci dessous)

## Un petit rappel sur la méthodologie proposée dans le cours :

1. Découvrir l'API
2. Requête l'API
3. Intégrer les requêtes SQL dans le RPG.

## Environnement

---

- P8 power8
- [Http Server Administration](#)
  - avec votre login Ibm i
- serveur WSERVICE2.
- le programme déployé est RPG4\*\*/LSTDVD
- la table utilisée est la table DVD de RPG4\*\*.
- le WS REST est LSTDVDREST\*\* (resource name) pour RPG4\*\*.
- l'utilisateur de votre service sera votre login IBMi ==> RPG4\*\*

## Liste des producteurs.

---

### Objectifs.

Utiliser getactiv.sql et lstdvd.sqlrpgle pour bâtir un programme qui consomme le service REST LSTDVDREST\*\* pour afficher à l'écran le titre et l'année des 10 premiers DVDs. en utilisant la démarche suivante :

1. Découvrir l'API
2. Requête l'API
3. Intégrer les requêtes SQL dans le RPG.

### Découvrir l'API.

Utiliser le fichier swagger pour découvrir l'API en s'aidant éventuellement d'outil.

- SOAPUI (?)
    - cf TP\_IntroServiceWeb
  - dans VSC avec l'extension [swagger viewer](#) à noter en l'état cette méthode ne permet pas d'appeler le service via l'IHM, mais rien n'empêche de récupérer les requêtes au format curl, puis de les lancer depuis QP2TERM ou dans le terminal passe de vsc.
1. Ajout du fichier swagger ou OPENAPI dans votre projet.
- afficher le fichier swagger du service REST LSTDVDREST\*\* via la console d'admin  
[Http Server Administration](#)

**Manage Deployed Services**

Data current as of 28 juin 2023 17:06:56.

Find service:

| Service name ^ | Status  | Type | Startup type | Service definition      |
|----------------|---------|------|--------------|-------------------------|
| ConvertTemp    | Running | SOAP | Automatic    | View WSDL               |
| LSTDVDRESTVY   | Running | REST | Automatic    | View Swagger <i>Uic</i> |
| LSTDVDSOAPVY   | Running | SOAP | Automatic    | View WSDL               |
| PRODUCTEUR     | Running | REST | Automatic    | View Swagger            |
| PRODUCTREST    | Running | REST | Automatic    | View Swagger            |
| PRODUCTSOAP    | Running | SOAP | Automatic    | View WSDL               |

Total 6 items

Swagger Viewer - Visual Studio | HTTP Server Administration on 1 | power8:2001/HTTPAdmin/Swagg x +

Non sécurisé | power8:2001/HTTPAdmin/SwaggerViewer?WebServiceName=LSTDVDRESTVY

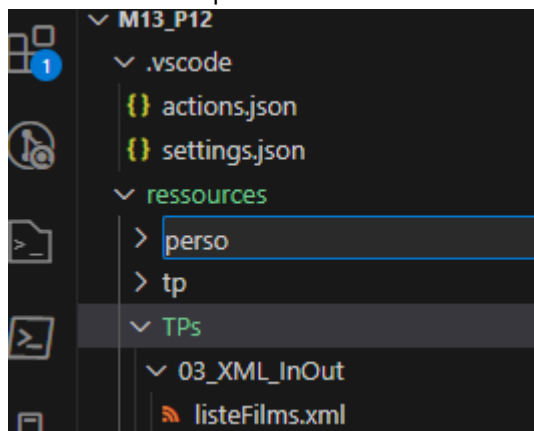
```
{
  "swagger": "2.0",
  "info": {
    "title": "LSTDVDRESTVY APIs",
    "description": "APIs available for LSTDVDRESTVY",
    "version": "1.0.0"
  },
  "host": "power8:10014",
  "schemes": [ "http" ],
  "basePath": "/web/services/LSTDVDRESTVY",
  "tags": [
    {
      "name": "LSTDVDRESTVY APIs",
      "description": "APIs available for LSTDVDRESTVY"
    }
  ],
  "definitions": {
    "DVD_Item": {
      "type": "object",
      "properties": {
        "titre": {
          "type": "string",
          "maxLength": 40
        },
        "annee": {
          "type": "number"
        }
      }
    },
    "LSTDVDResult": {
      "type": "object",

```

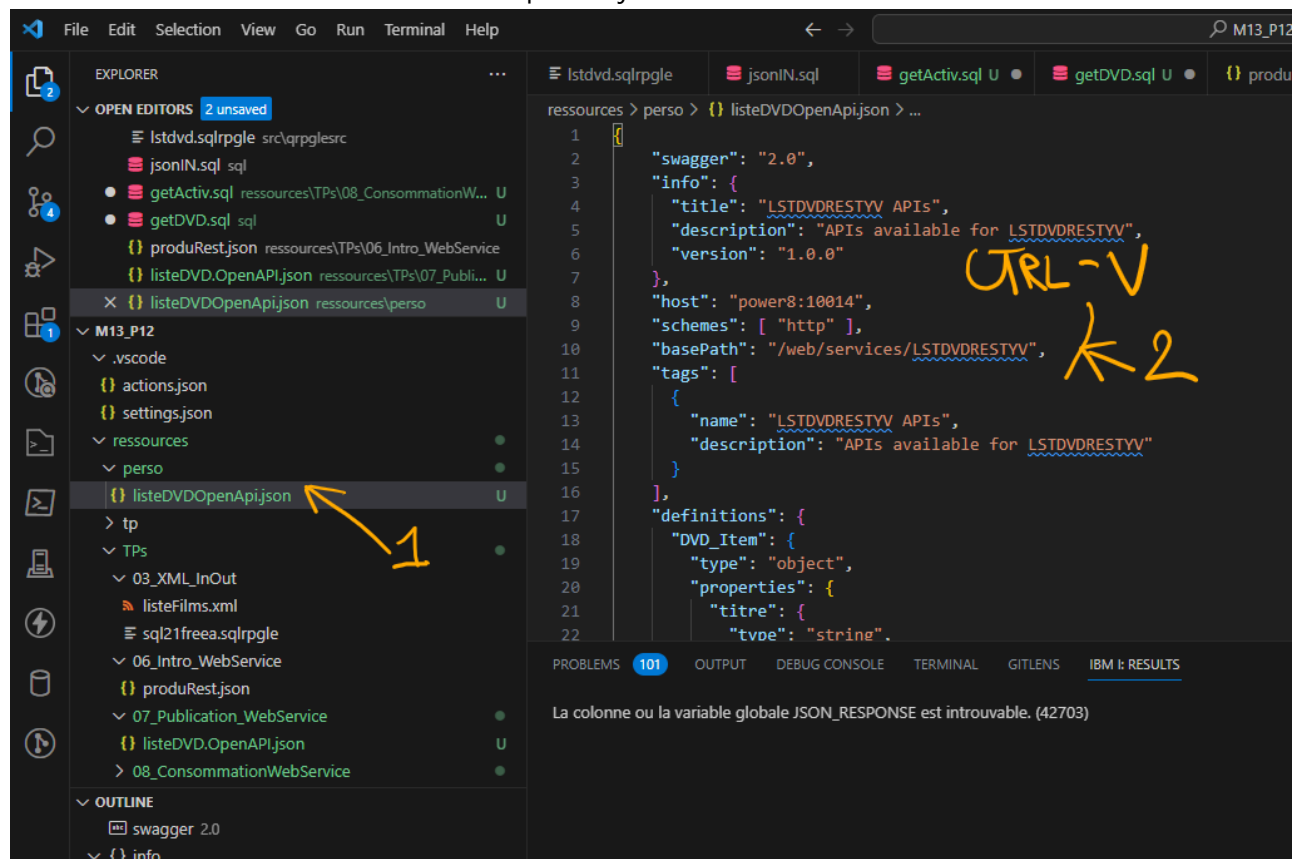
*URL-A  
Puio  
URL-C*

*①*

- creer un dossier perso dans le dossier ressources de votre projet.

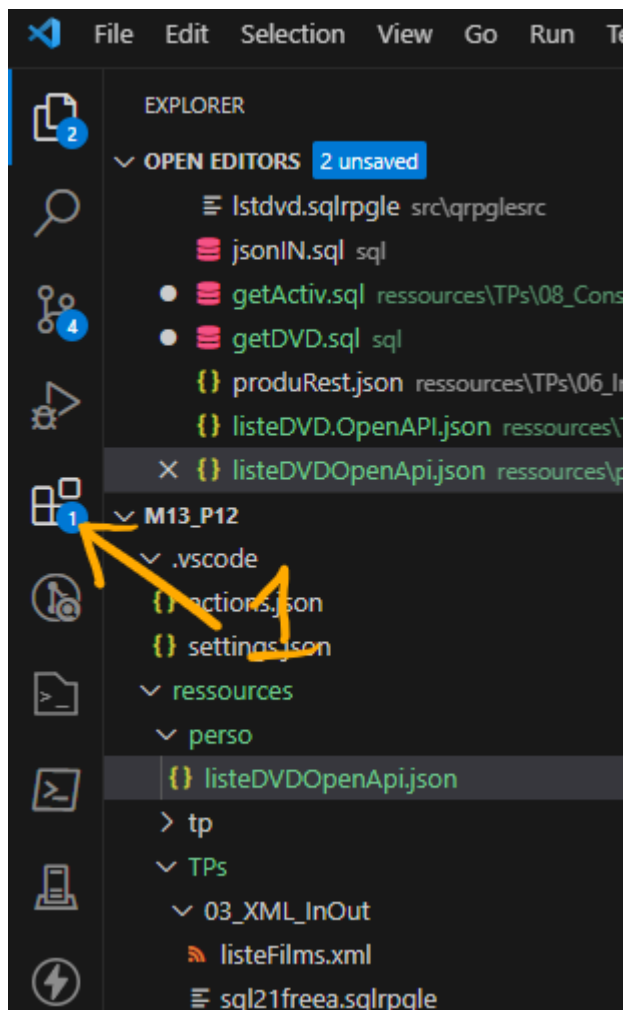


- dans ce dossier créer un fichier listeDVDOpenAPI.json

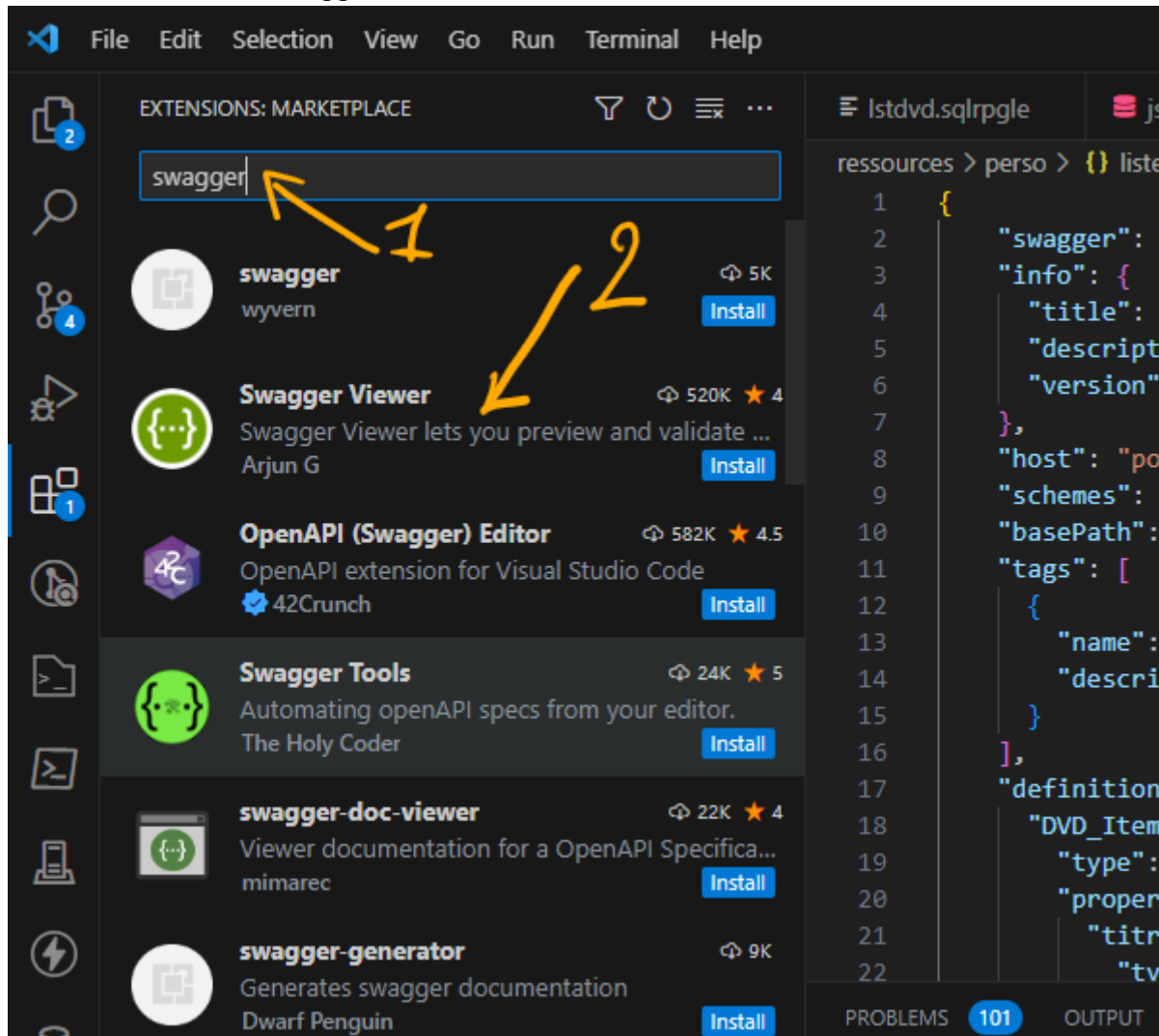


1. Installation de l'extension [swagger viewer](#)

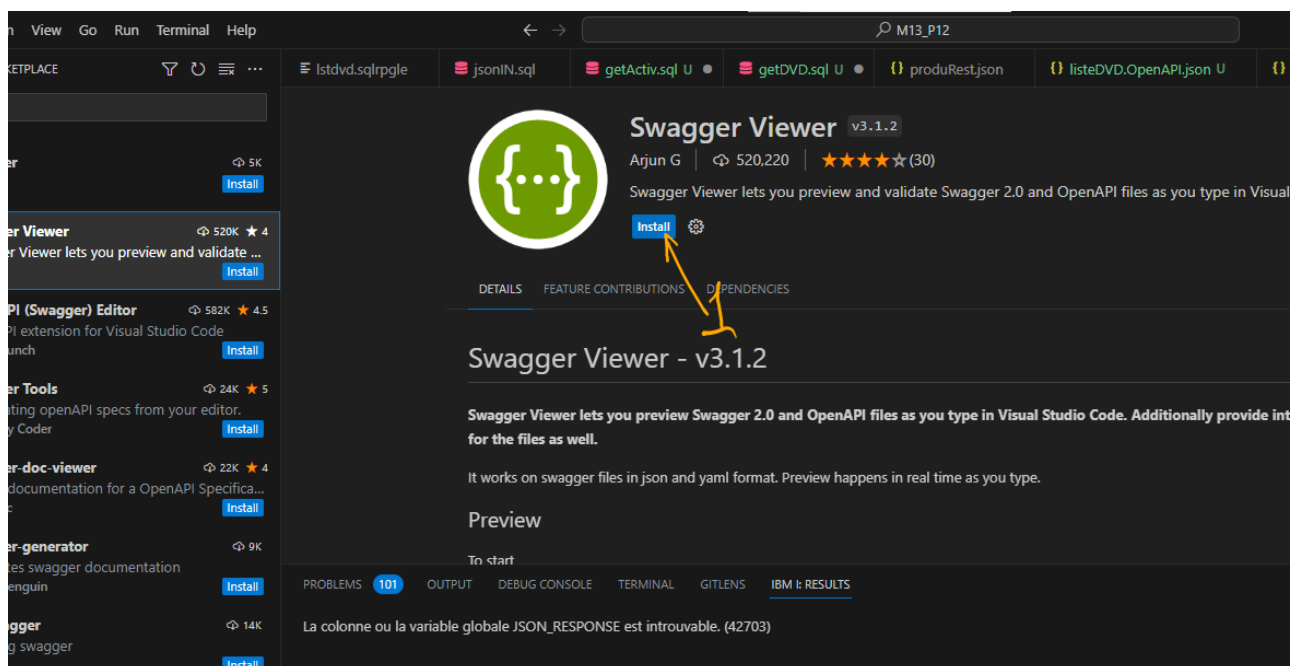
- ouvrir le volet extension de vsc



- rechercher l'extension swagger viewer

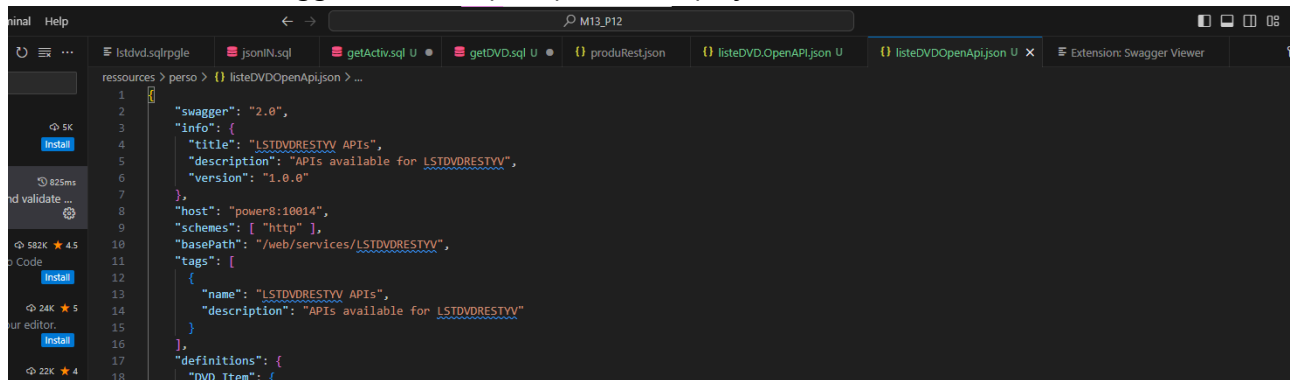


- installer l'extension



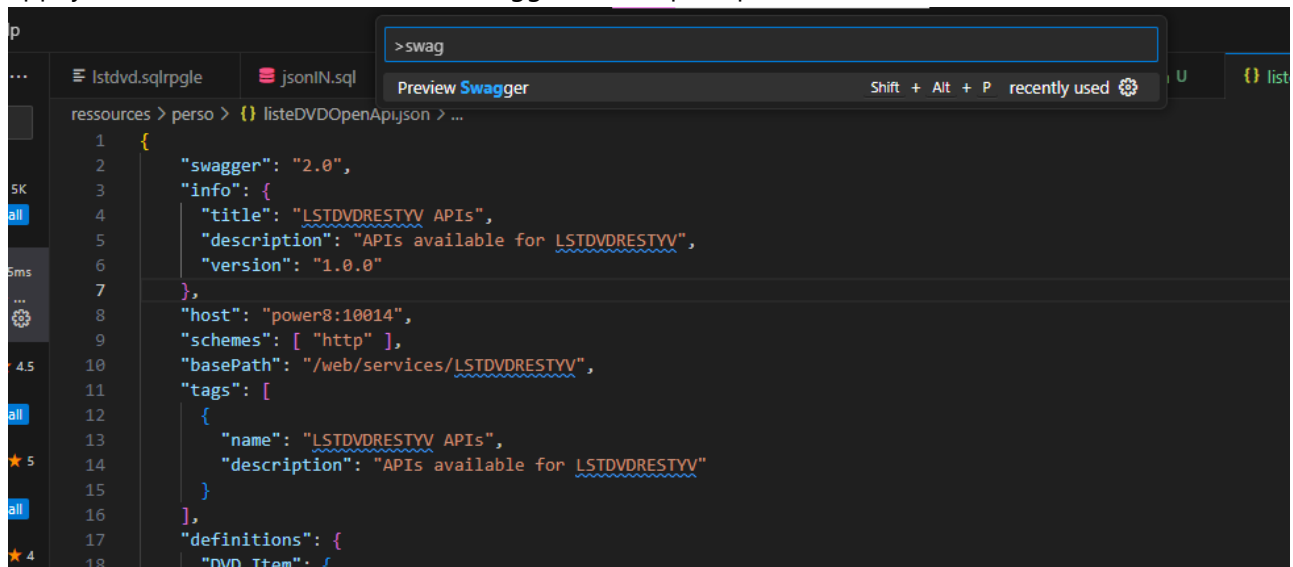
- affichage du swagger ui

- ouvrir votre fichier swagger listeDVDOpenApi de votre projet.



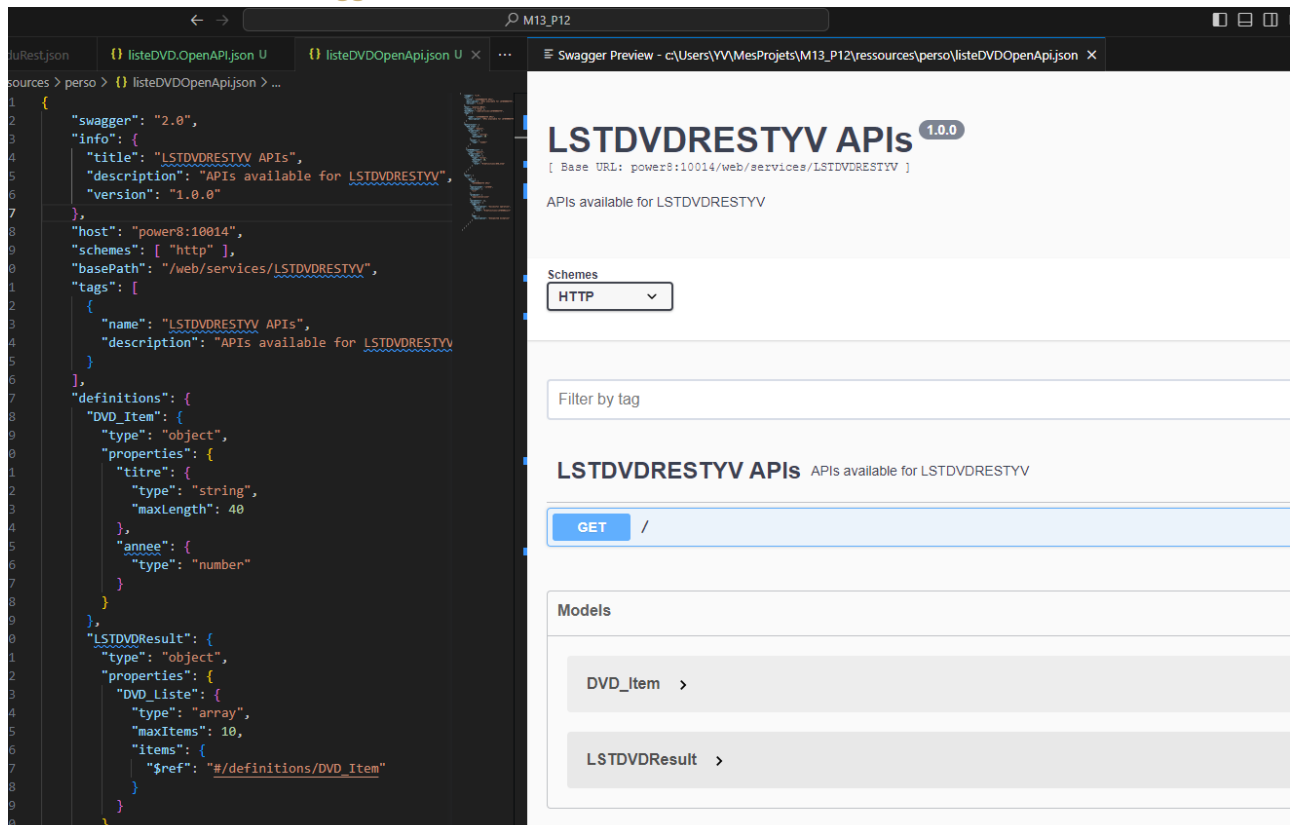
```
1 {
2   "swagger": "2.0",
3   "info": {
4     "title": "LSTDVDRESTYV APIs",
5     "description": "APIs available for LSTDVDRESTYV",
6     "version": "1.0.0"
7   },
8   "host": "power8:10014",
9   "schemes": [ "http" ],
10  "basePath": "/web/services/LSTDVDRESTYV",
11  "tags": [
12    {
13      "name": "LSTDVDRESTYV APIs",
14      "description": "APIs available for LSTDVDRESTYV"
15    }
16  ],
17  "definitions": {
18    "DVD_Item": {
```

- appuyer sur la touche F1 et inscrivez swagger dans le prompt

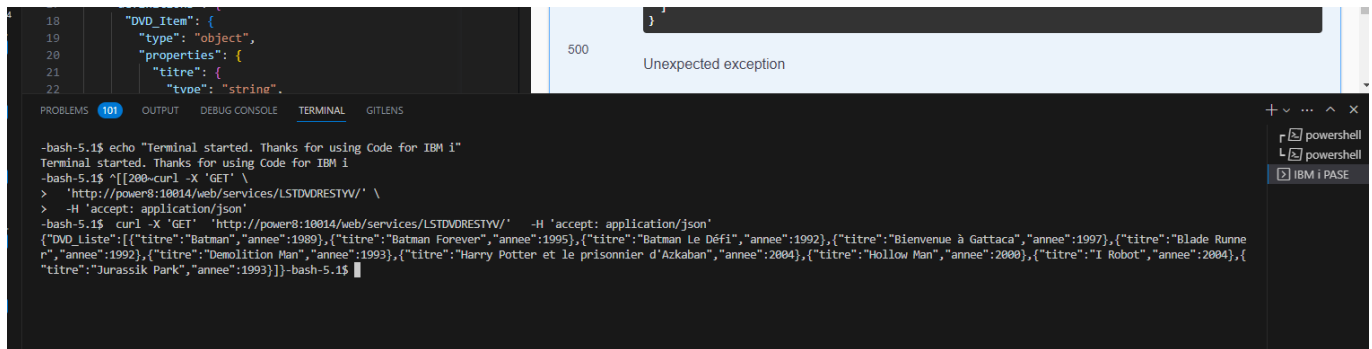


```
> swagger
Preview Swagger
Shift + Alt + P recently used
```

- selectionner Preview swagger



The Swagger Preview window displays the API information for LSTDVDRESTYV APIs 1.0.0. The base URL is power8:10014/web/services/LSTDVDRESTYV. The API is available for LSTDVDRESTYV. The Schemes dropdown is set to HTTP. The Filter by tag field is empty. The LSTDVDRESTYV APIs section shows the GET / endpoint. The Models section lists DVD\_Item and LSTDVDResult.



```
18 "DVD_Item": {
19   "type": "object",
20   "properties": {
21     "titre": {
22       "type": "string",
23     }
24   }
25 }
26 }
```

500  
Unexpected exception

PROBLEMS 101 OUTPUT DEBUG CONSOLE TERMINAL GIT LENS

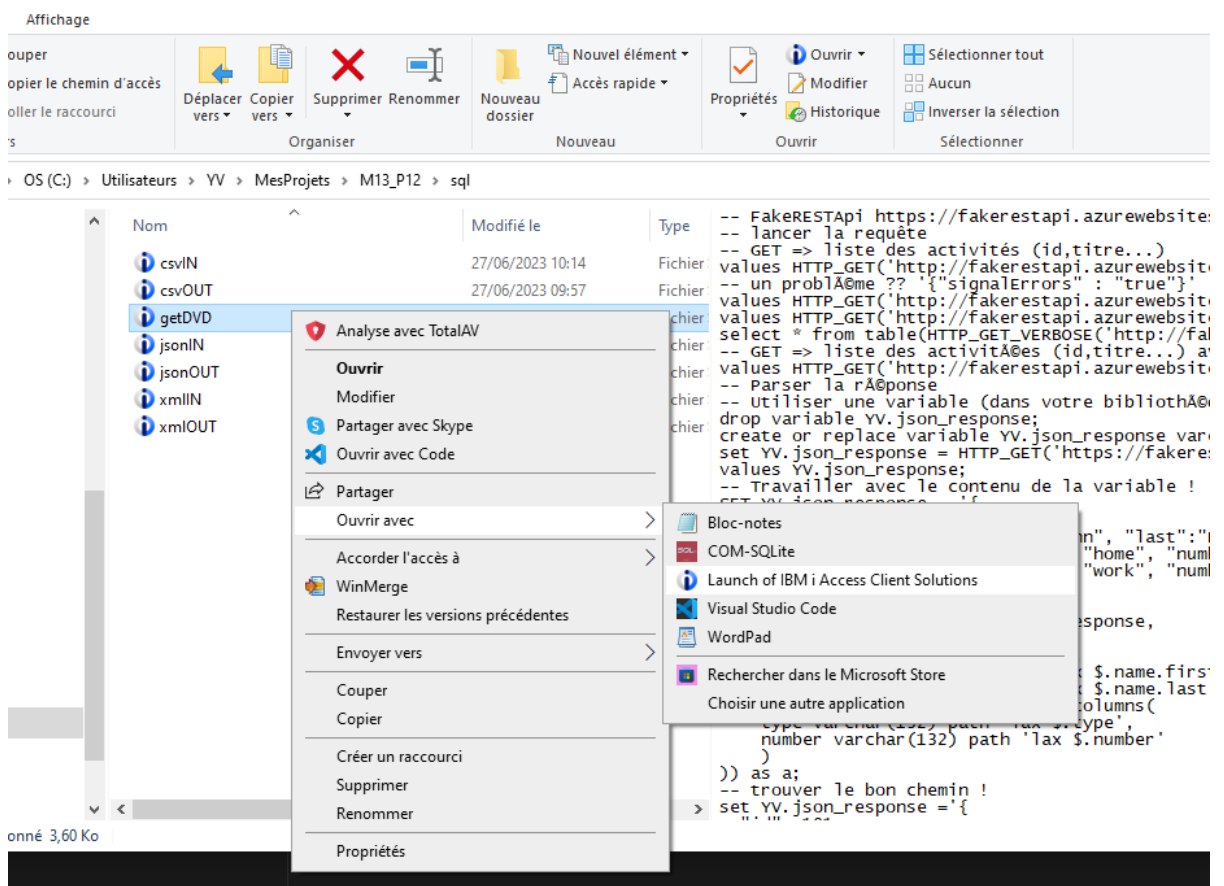
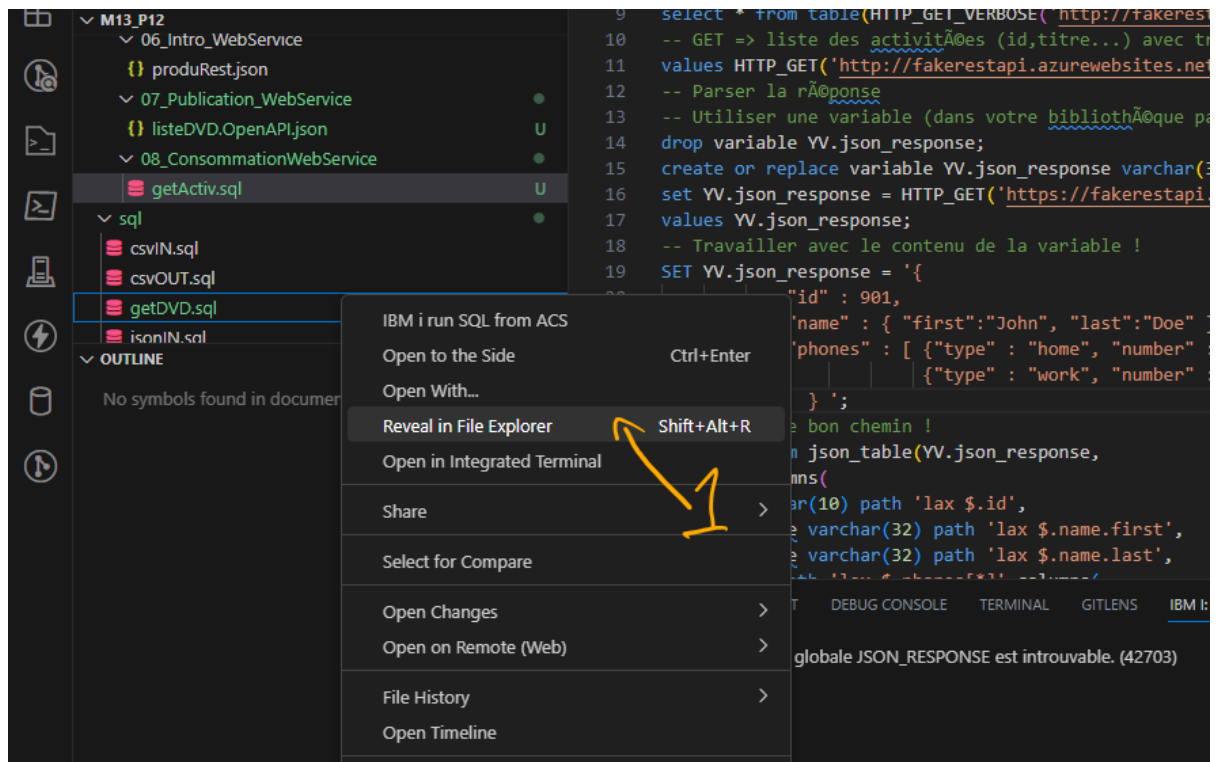
```
-bash-5.1$ echo "Terminal started. Thanks for using Code for IBM i"
Terminal started. Thanks for using Code for IBM i
-bash-5.1$ curl -X 'GET' \
> 'http://power8:10014/web/services/LSTDVDRESTVW/' \
> -H 'accept: application/json'
-bash-5.1$ curl -X 'GET' 'http://power8:10014/web/services/LSTDVDRESTVW/' -H 'accept: application/json'
{"DVD_liste":[{"titre":"Batman","annee":1989}, {"titre":"Batman Forever","annee":1995}, {"titre":"Batman Le D fi","annee":1992}, {"titre":"Bienvenue   Gattaca","annee":1997}, {"titre":"Blade Runne
r","annee":1992}, {"titre":"Demolition Man","annee":1993}, {"titre":"Harry Potter et le prisonnier d'Azkaban","annee":2004}, {"titre":"Hollow Man","annee":2000}, {"titre":"I Robot","annee":2004}, {"
"titre":"Jurassik Park","annee":1993}]}-bash-5.1$
```

## Requ ter l'API

En vous inspirant du script sql getActiv.sql (ressources\TPs\08\_ConsommationWebService\getActiv.sql)

- Cr ez un script sql (getDVD.sql) permettant de :
  - r aliser un GET sur LSTDVDREST\*\* pour lister les 10 premiers DVDs.
  - Sauvegardez ce script et conservez le dans votre projet.A noter Tester ce script dans aCS, vous pouvez l' diter dans ACs et le sauvegarder dans ACS.





# 1. Définir la requête HTTP à jouer pour accéder au Web Service.

- Sélectionner l'instruction en fonction du verbe HTTP requis. [HTTP\\_GET](#)

```
values HTTP_GET('http://fakereapi.azurewebsites.net/api/v1/Activities'
, '');
```

avec l'url requise.

- vérifier que le résultat correspond à vos attentes.
- ajouter la gestion des erreurs  

```
{"signalErrors" : "true"}
```

 pour tester mettez une mauvaise url !  
 la fonction [HTTP\\_GET\\_VERBOSE](#) vous permet d'analyser le problème.
- préciser que vous utilisez le format json pour échanger avec le serveur.

```
{
  "header": "Accept,application/json",
  "header": "Content-Type,application/json",
  "signalErrors" : "true"
}
```

2. parser la réponse au format json se trouvant dans la résultat.

Une bonne pratique est d'utiliser une variable SQL de travail.

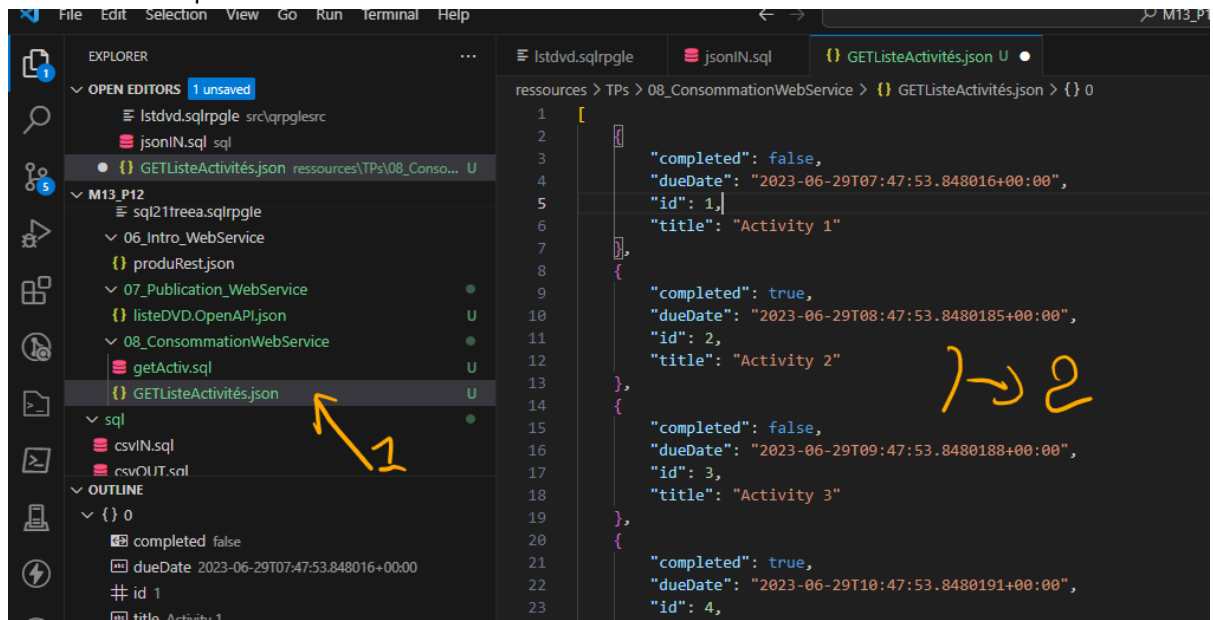
Cela permet de travailler sans appeler le Web Service (hors connexion , phase de développement)

- Créer et Utiliser une variable (dans votre bibliothèque par exemple RPG4\*\*)
- Placer le résultat de la requête dans la variable.
- Visualiser le contenu de la variable.
- Travailler avec le contenu de la variable , pour trouver le bon chemin!

Nous allons utiliser l'instruction [json\\_table](#)

- définir la référence ==> la ligne de la future table.

ce que j'aime faire c'est visualiser le json dans vsc en créant dans mon projet un fichier json contenant la réponse.



```

1  [
2      {
3          "completed": false,
4          "dueDate": "2023-06-29T07:47:53.848016+00:00",
5          "id": 1,
6          "title": "Activity 1"
7      },
8      {
9          "completed": true,
10         "dueDate": "2023-06-29T08:47:53.8480185+00:00",
11         "id": 2,
12         "title": "Activity 2"
13     },
14 ]

```

'\$'

- définir les zones de la table et les attributs correspondants dans le json.

```

1  [
2      {
3          "completed": false,
4          "dueDate": "2023-06-29T07:47:53.848016+00:00",
5          "id": 1,
6          "title": "Activity 1"
7      },
8      {
9          "completed": true,
10         "dueDate": "2023-06-29T08:47:53.8480185+00:00",
11         "id": 2,
12         "title": "Activity 2"
13     },
14     {
15         "completed": false,
16         "dueDate": "2023-06-29T09:47:53.8480188+00:00",
17         "id": 3,
18         "title": "Activity 3"
19     }
20 ]

```

```

columns(
code integer path '$.id',
titre varchar(132) path '$.title'
)

```

- tester
- définir une selection, tri, calcul.... en sql sur la table résultante.
- ☹ même le CTRL-SPACE d'ACS fonctionne.

\*getActiv.sql

```

23 -- Travailler avec le contenu de la variable , pour trouver le bon chemin!
24 -- trouver la ligne de la future table
25 -- trouver les colonnes
26 select * from json_table(YV.json_response,
27 '$' columns(
28   code integer path '$.id',
29   titre varchar(132) path '$.title'
30 )) as a;
31 -- définir une selection,tri,calcul.... en sql sur la table résultante.
32 select * from json_table(YV.json_response,
33 '$' columns(
34   code integer path '$.id',
35   titre varchar(132) path '$.title'
36 )) as a where code > 10 order by ;
37 -- les deux ensembles
38 SELECT *
39   FROM JSON_TABLE(
40     HTTP_GET('https://fak
41     '$'
42     COLUMNS(
43       id INTEGER PATH '$.
44       titre VARCHAR(132)
45     )
46   ) AS a
47   ORDER BY id
48   FETCH FIRST 2 ROWS ONLY.

```

Toutes les colonnes

- a.code INTEGER
- a.titre VARCHAR(132)

| CODE | TITRE      |
|------|------------|
| 1    | Activity 1 |
| 2    | Activity 2 |
| 3    | Activity 3 |

3. mixer la requête HTTP et l'analyse du résultat en une seule requête.

```

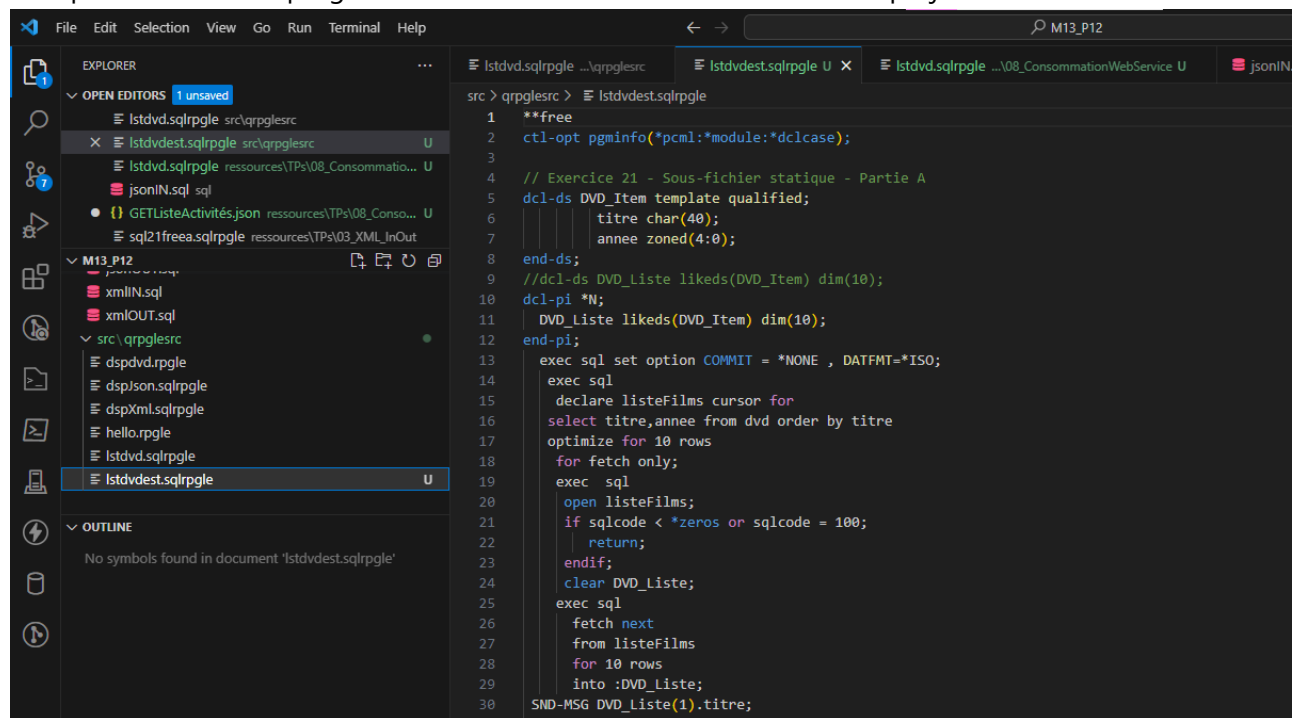
37 -- mixer la requête HTTP et l'analyse du résultat en une seule requête.
38 SELECT *
39   FROM JSON_TABLE(
40     HTTP_GET(
41       'http://fakereapi.azurewebsites.net/api/v1/Activities',
42       '{"header": "Accept,application/json","header": "Content-Type,application/json","signalErrors" : "true"}'),
43     '$'
44     COLUMNS(
45       code INTEGER PATH '$.id', titre VARCHAR(132) PATH '$.title'
46     )
47   ) AS a
48   WHERE code > 10
49   ORDER BY a.titre;
50

```

| CODE | TITRE       |
|------|-------------|
| 11   | Activity 11 |
| 12   | Activity 12 |
| 13   | Activity 13 |
| 14   | Activity 14 |
| 15   | Activity 15 |
| 16   | Activity 16 |
| 17   | Activity 17 |
| 18   | Activity 18 |
| 19   | Activity 19 |
| 20   | Activity 20 |
| 21   | Activity 21 |

Intégrer les requêtes SQL dans le RPG.

1. Recopiez le source du programme LSTDVD en LSTDVDREST dans votre projet.



The screenshot shows an IDE with the following components:

- EXPLORER:** Displays a project structure with folders like 'src' and 'resources'. The file 'Lstdvddest.sqlrpgle' is selected under the 'src' folder.
- OPEN EDITORS:** Shows several open files, including 'Lstdvddest.sqlrpgle' and 'Lstdvd.sqlrpgle'.
- EDITOR:** Displays the source code of 'Lstdvddest.sqlrpgle'. The code is in FreeForm SQL and includes comments and SQL statements. The visible code is as follows:

```
1  **free
2  ctl-opt pgminfo(*pcml:*module:*dclcase);
3
4  // Exercice 21 - Sous-fichier statique - Partie A
5  dcl-ds DVD_Item template qualified;
6      titre char(40);
7      annee zoned(4:0);
8  end-ds;
9  //dcl-ds DVD_Liste likeds(DVD_Item) dim(10);
10 dcl-pi *N;
11   DVD_Liste likeds(DVD_Item) dim(10);
12 end-pi;
13
14 exec sql set option COMMIT = *NONE , DATFMT=*ISO;
15
16 exec sql
17   declare listeFilms cursor for
18   select titre,annee from dvd order by titre
19   optimize for 10 rows
20   for fetch only;
21   exec sql
22     open listeFilms;
23     if sqlcode < *zeros or sqlcode = 100;
24       return;
25     endif;
26     clear DVD_Liste;
27   exec sql
28     fetch next
29     from listeFilms
30     for 10 rows
31     into :DVD_Liste;
32   SMD-MSG DVD_Liste(1).titre;
```

2. Modifiez le source LSTDVDREST pour y intégrer la requête que vous avez préalablement définie dans getdvd.sql.

The first screenshot shows two panels. The left panel displays an RPG program named `lststdvdrest.sqlrpgle` with the following code:

```

1 **free
2 cti-opt pgminfo(*pcml:'module:*dclcase');
3
4 // Exercice 21 - Sous-fichier statique - Partie A
5 dcl-ds DVD_Item template qualified;
6     titre char(40);
7     annee zoned(4:0);
8 end-ds;
9 //dcl-ds DVD_Liste likeds(DVD_Item) dim(10);
10 dcl-pi *N;
11     DVD_Liste likeds(DVD_Item) dim(10);
12 end-pi;
13 exec sql set option COMMIT = *NONE , DATFMT=*ISO;
14 exec sql
15     declare listeFilms cursor for
16     select titre,annee from dvd order by titre
17     optimize for 10 rows
18     for fetch only;
19 exec sql
20     open listeFilms;
21     if sqlcode < *zeros or sqlcode = 100;
22         return;
23     endif;
24     clear DVD_Liste;
25 exec sql
26     fetch next
27     from listeFilms
28     for 10 rows
29     into :DVD_Liste;
30 SND-MSG DVD_Liste(1).titre;
31 SND-MSG DVD_Liste(10).titre;
32 *inlr = *on;
33
34

```

The right panel shows a SQL query named `getDvd.sql`:

```

1 sql> getDvd.sql
2 // $DVD_Liste' columns(
28     annee integer path '$.annee',
29     titre varchar(132) path '$.titre'
30 )) as a;
31 -- définir une sélection,tri,calcul... en sql sur la table résultante.
32 select * from json_table(VV.json_response,
33     '$.DVD_Liste' columns(
34         annee integer path '$.annee',
35         titre varchar(132) path '$.titre'
36     )) as a order by a.titre;
37 -- mixer la requête HTTP et l'analyse du résultat en une seule requête.
38 SELECT *
39 FROM JSON_TABLE(
40     HTTP_GET(
41         'http://power8:10014/web/services/LSTDVDRESTVW/',
42         '{"header": "Accept,application/json","header": "Content-Type,application/json","signalErrors": "true"}'),
43     '$.DVD_Liste'
44     COLUMNS(
45         annee INTEGER PATH '$.annee', titre VARCHAR(132) PATH '$.titre'
46     )
47 ) AS a
48 ORDER BY a.titre;
49

```

The second screenshot shows the same code with some additional annotations and a different line numbering for the SQL query.

### 3. tester en appelant le programme en 5250

The screenshot shows a 5250 terminal window with the following output:

```

> call lststdvdrest
Pointeur non défini pour position mémoire référencée.
(C G D F) Erreur de paramètre ou de pointeur.
? C
(C G D F) Erreur de paramètre ou de pointeur.
? C
Application error. MCH3601 unmonitored by LSTDVDREST at statement
00000000134, instruction X'0000'.
> CALL PGM(LSTDVDREST) PARM((' '))
Batman
Jurassik Park

```

The text "A suivre" is visible in the bottom right corner of the terminal window.

### 4. Appeler l'appel de ce programme LSTDVDREST en lieu et place de LSTDVD dans le programme DSPDVD et tester !

