Manual

Se hizo una base de datos de videojuegos cargandolos desde la api de twitch, para poder tener guardados todos los datos de los videojuegos, junto con sus características más importantes, como géneros, franquicias, modos de juego, compañías, fechas de lanzamiento y todo lo relacionado al propio videojuego.

Paso 1: Definir la bases de datos

Antes de empezar a ver la construcción de la propia base de datos, primero se debe de ver cuáles serán los datos que se manejan en la propia, cuáles serán los datos que nos interesan y los que nos pide el cliente.

Se solicitó que la base de datos contuviera los datos que muestra la página principal de cada juego (ej. Super Smash Bros. Melee (2001) (igdb.com)), con los que luego se mostrarán los juegos individuales con sus respectivos datos generales y específicos entre todo lo referente a los videojuegos.

Para saber de qué manera serán los datos, se utilizará la api de twitch Getting Started – IGDB API docs, la cual nos indicará cómo recoger los datos y cuál será su naturaleza

INFORMATION

Edit Game Information

IGDB ID: 1627

Release Dates:

Nintendo GameCube
2001-11-21 - **Japan [JP]**
2001-12-3 - **North America [NA]**
2002-5-24 - **Europe [EU]**
2002-5-31 - **Australia [AU]**
TBD - **Brazil [BR]**

Developers:

HAL Laboratory

Publishers:

Nintendo
Gradiente

Game Modes:

Co-operative
Multiplayer
Single player

Multiplayer Modes:

Nintendo GameCube:

- Offline Co-op max players: 3
- Online Co-op max players:
- Online max players:
- Offline max players: 4
- Offline Co-op ✔
- Online Co-op ✘
- LAN Co-op ✘
- Co-op Campaign ✘
- Online Split-Screen ✘
- Offline Split-Screen ✔
- Drop in/out ✘

Genre:

Fighting
Platform

Themes:

Action

Series:

Super Smash Bros.

Paso 2: Diseñar la Estructura de la Base de Datos

Luego de ya saber todos los datos que se necesitarán, se prosigue con la creación del modelo entidad--relación con el cual definir las relaciones entre tablas.

Inicialmente se hizo una diagrama entidad-relación con el cual se hicieron tablas con llave primaria “cheksum”, que posteriormente al utilizar y manejar la APi nos daríamos cuenta que las propias tablas ya vienen numeradas con sus propios id’s, y de igual manera, en lo que avanzaba el tiempo, el cliente pidió algunos otros datos, con lo que tuvimos que agregar algunas relaciones y tablas extras.

Diagrama Inicial

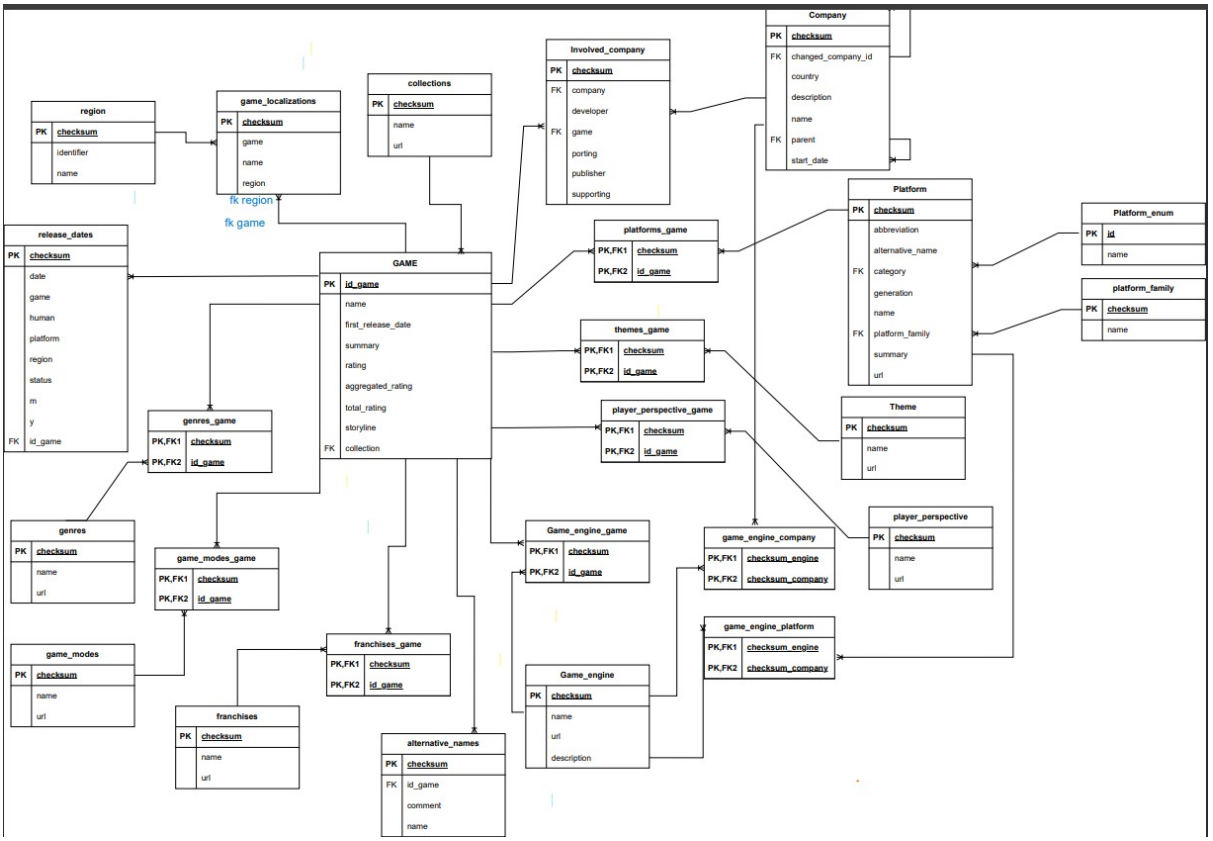
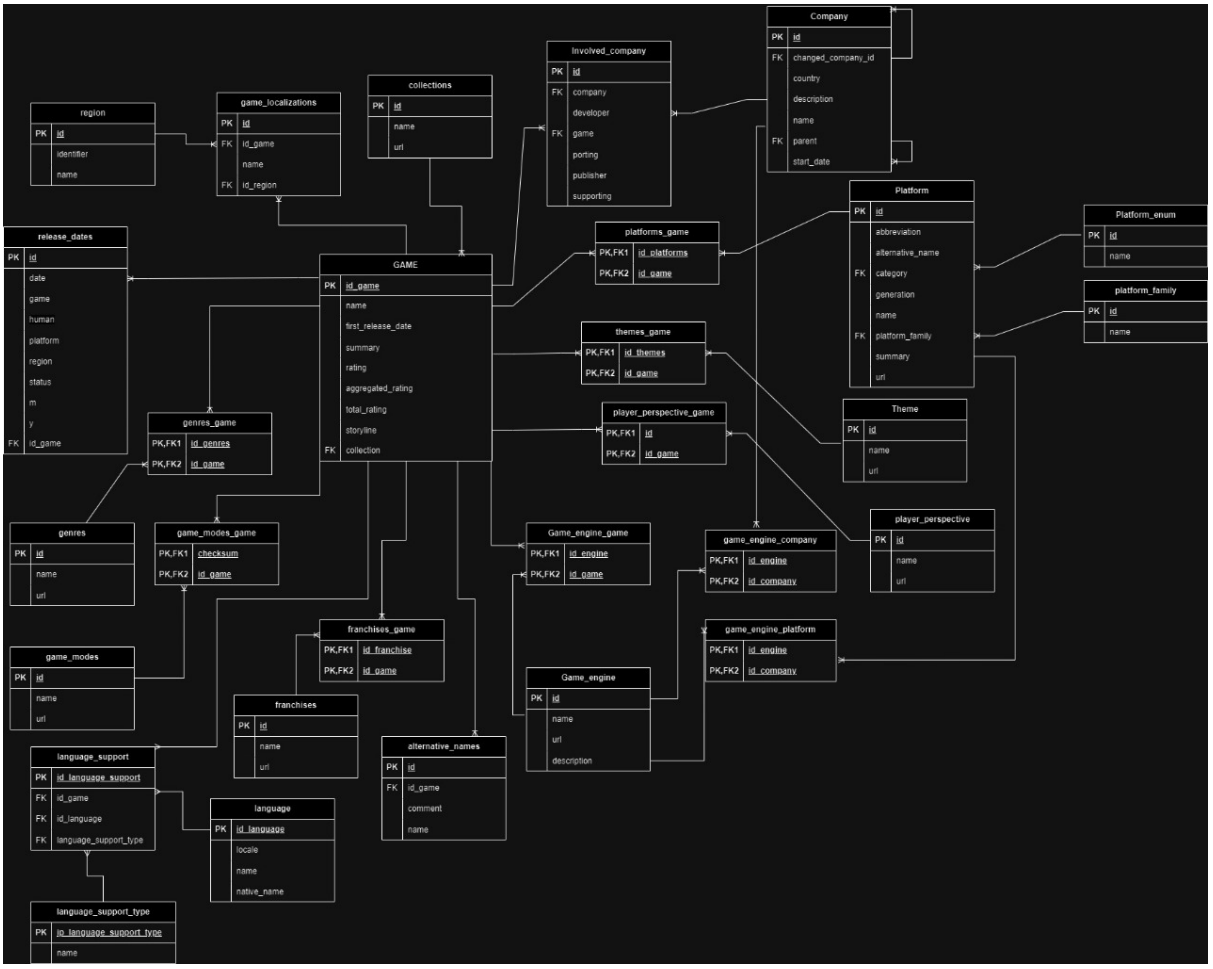


Diagrama Final



Paso 3: Crear la Base de Datos

Se decidió utilizar el DBMS de SQL Server para poder manejar los datos. La base se creó en la nube de AWS RDS, se utilizó una base del tamaño t2.micro, que posteriormente se cambiaría a una t3.medium, ya que las consultas y la inserción de datos tomaban demasiado tiempo, al ser demasiados datos.

Luego de haber definido el diagrama entidad-relación, se procede a crear las tablas, junto con sus datos, sus llaves, primarias y foráneas.
Las tablas que se decidieron crear fueron:

- **game**
- **genre**
- **game_genre**
- **game_mode**
- **game_game_mode**
- **collection**
- **release_date**
- **game_localization**
- **region**

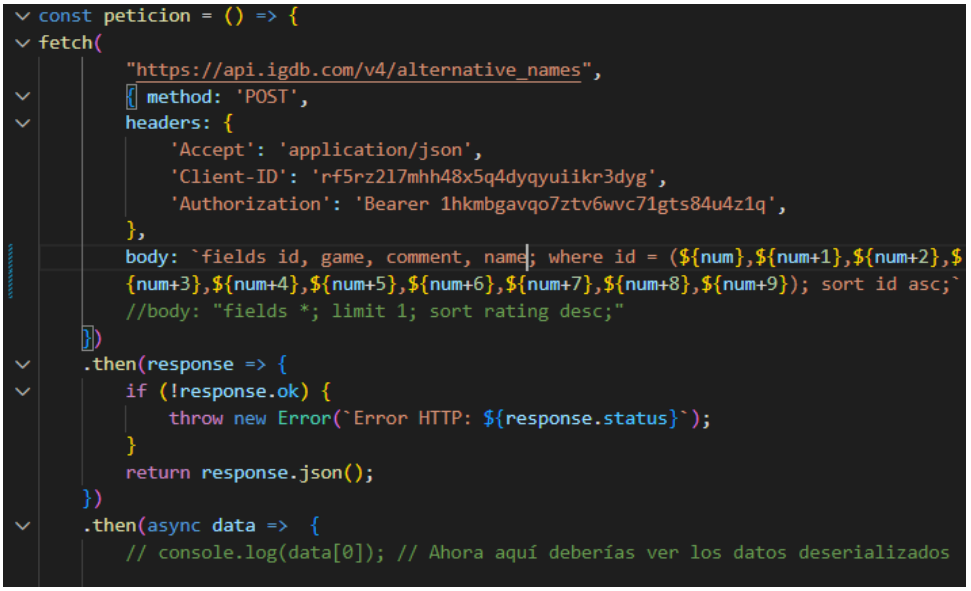
- **platform**
- **game_platform**
- **theme**
- **game_theme**
- **player_perspective**
- **game_player_perspective**
- **game_engine**
- **game_engine_platform**
- **company**
- **involved_company**
- **game_engine_company**
- **language**
- **language_support**

Entre estas tablas, la mayoría se mandan a cargar individualmente, lo que indica que son independientes de otras, y entre ellas y la tabla principal “game” se tienen tablas intermedias que nos sirven para guardar la referencia entre el juego y los detalles específicos sobre la otra tabla a la que estará relacionada.

Paso 4: Desarrollar la API

Se utilizó una API en Javascript, en la cual consumimos la API de twitch, anteriormente mencionada, en donde se hizo un archivo javascript para cada una de las tablas independientes.

Ejemplo de una llamada a la API de twitch y posteriormente de datos a la base de datos, en este caso será la tabla de “alternative_names”:



Como se puede observar, se mandan a llamar a los datos que se necesitarán para la tabla que insertaremos, se guardan en un arreglo, el cual posteriormente se iterará para verificar los demás datos e insertarlos de manera incremental.

Se utilizó un manejador de base de datos de javascript “mssql”, con el cual se hacía cómodo hacer las inserciones.

Paso 5. Inserción de datos

Para insertar datos a la base de datos se tuvo que hacer de 10 en 10 datos, debido a que la propia API de twitch nos restringe las consultas, por lo que se recorrió de manera ordenada y ascendente desde el número 1, hasta el id final de los datos que se mandarán a llamar.

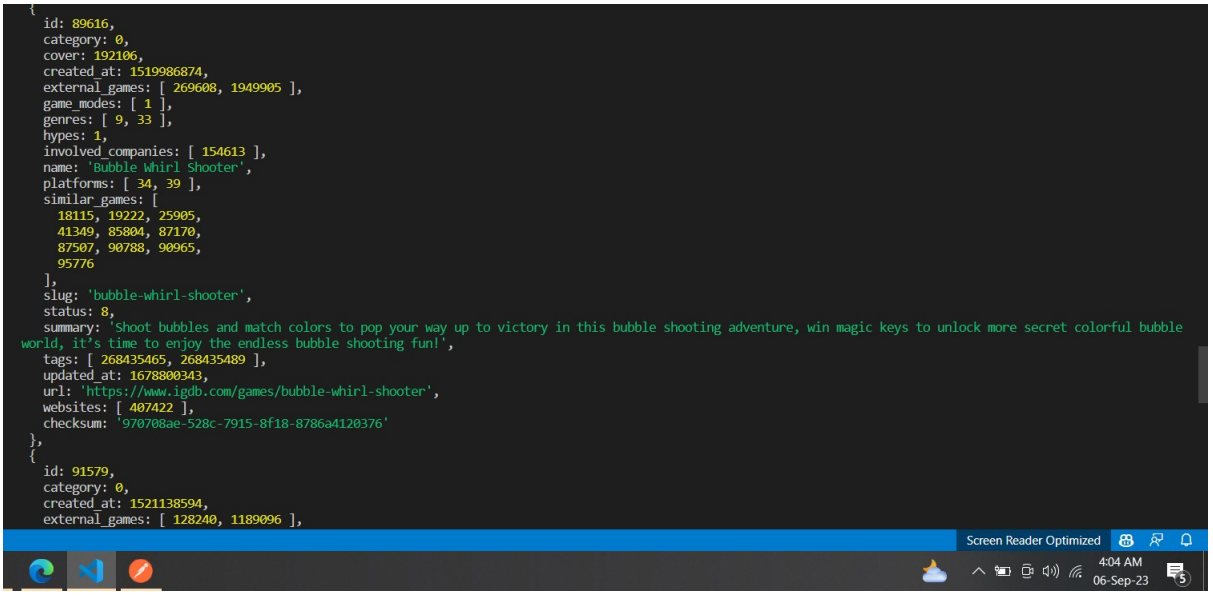
Descripción de endpoints importantes:

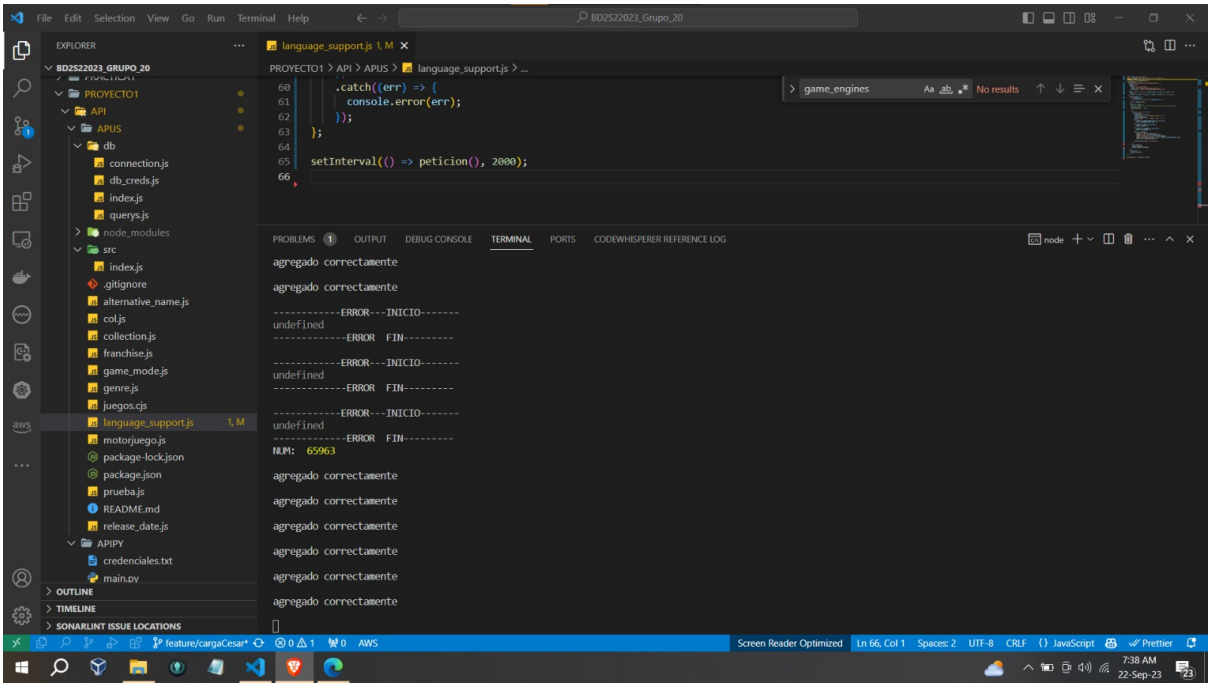
- “games”: Se utilizó el endpoint games, para mandar a llamar los datos de los juegos, y debido a que era la tabla que contiene más datos, se opto por realizar al final, debido a las inserciones de las tablas intermedias y así poder insertarlas junto con los propios juegos.

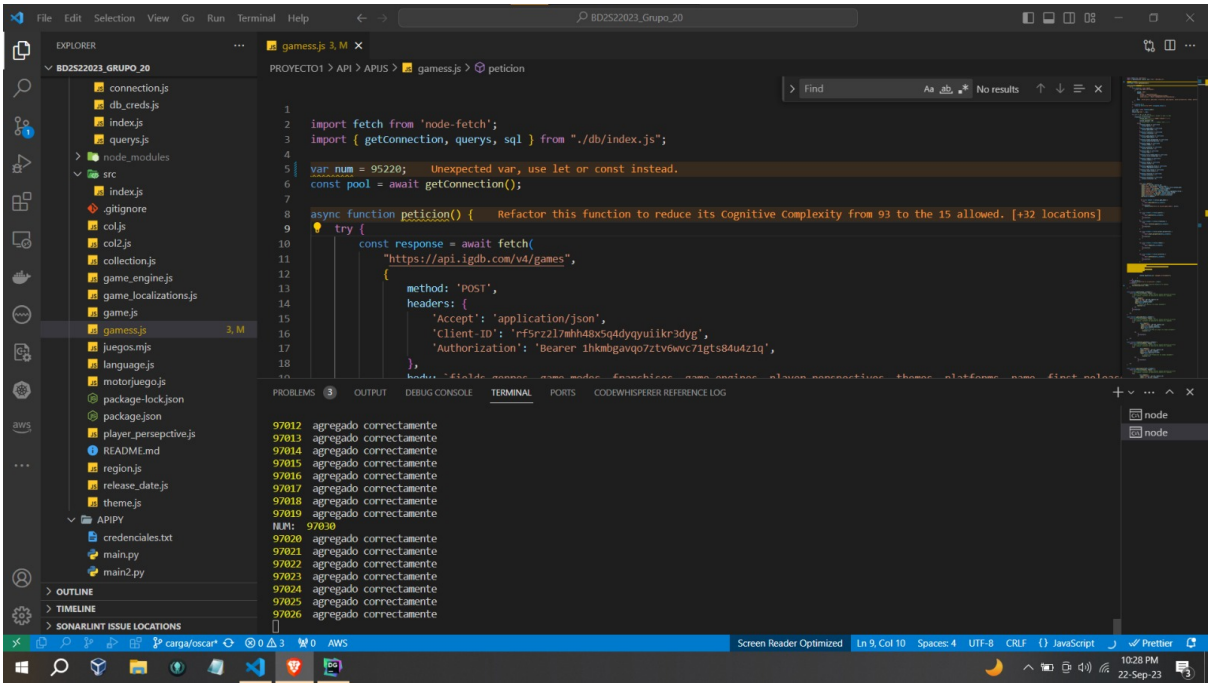
Con el manejador de “mssql” se hicieron las queries de las inserciones de manera aparte, pero la sintaxis es la propia de SQL Server

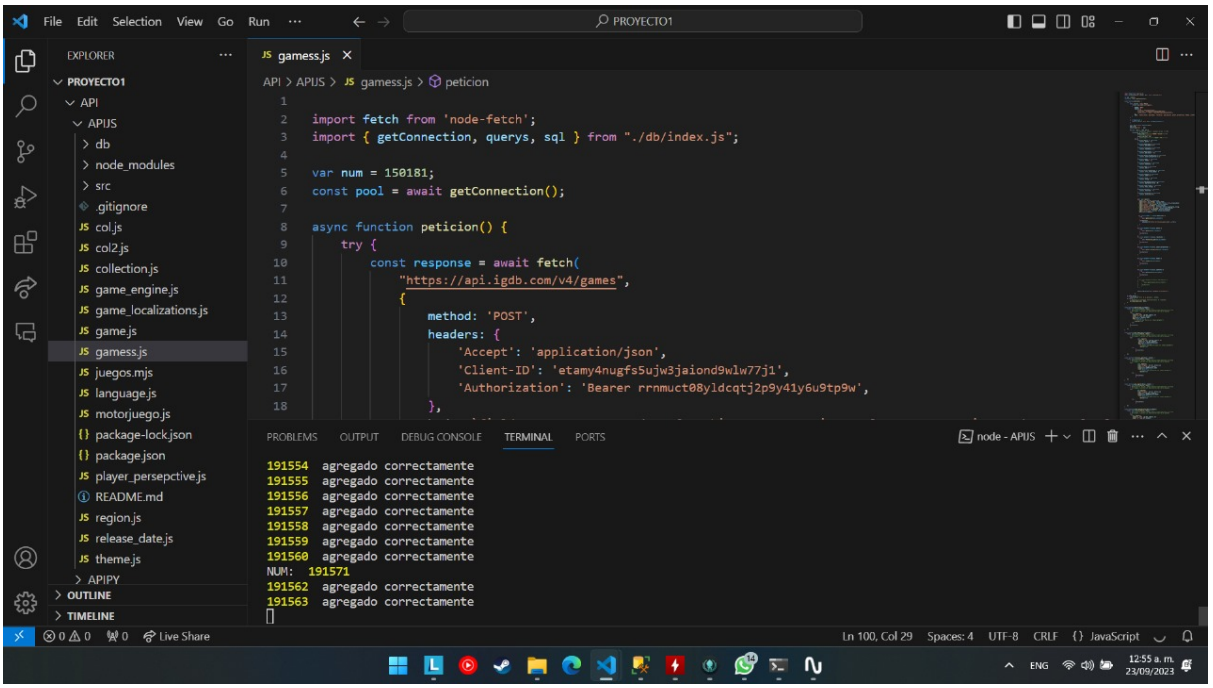
```
db > querys.js > querys > addInvolved_Company
export const querys = {

    addCollection: "INSERT INTO collections (id, name, url) VALUES(@id, @name, @url);",
    addRegion: "INSERT INTO region (id, identifier, nombre) VALUES(@id, @identifier, @name);",
    addTheme: "INSERT INTO theme (id, name, url) VALUES(@id, @name, @url);",
    addPlayer_Perspective: "INSERT INTO player_perspective (id, name, url) VALUES(@id, @name, @url);",
    addGame: "INSERT INTO game (id_game, name, first_release_date, summary, rating, aggregated_rating, local_rating, storyline, collections) VALUES(@id, @name, @first_release_date, @summary, @rating,
    @aggregated_rating, @local_rating, @storyline, @collections);",
    addGenre_Game: "INSERT INTO genres_game (id, id_game) VALUES(@id, @id_game);",
    addGame_Mode_Game: "INSERT INTO game_mode_game (id, id_game) VALUES(@id, @id_game);",
    addFranchise_Game: "INSERT INTO franchise_game (id, id_game) VALUES(@id, @id_game);",
    addGame_Engine_Game: "INSERT INTO game_engine_game (id, id_game) VALUES(@id, @id_game);",
    addPlayer_Perspective_Game: "INSERT INTO player_perspective_game (id, id_game) VALUES(@id, @id_game);",
    addThemes_Game: "INSERT INTO themes_game (id, id_game) VALUES(@id, @id_game);",
    addPlatform_Game: "INSERT INTO platform_game (id, id_game) VALUES(@id, @id_game);",

    addAlternative_Name: "INSERT INTO alternative_name (id, id_game, comment, name) VALUES(@id, @id_game, @comment, @name);",

    addGame_Localization: "INSERT INTO game_localization (id, id_game, name, id_region) VALUES(@id, @id_game, @name, @id_region);",

    addInvolved_Company: "INSERT INTO involved_company (id, company, game, developer, porting, publisher, supporting) VALUES(@id, @company, @game, @developer, @porting, @publisher, @supporting);",

    addengine: "INSERT INTO game_engine (id, name, url, description) VALUES(@id, @name, @url, @description);",
    addengine_company: "INSERT INTO game_engine_company (id_engine, id_company) VALUES(@id_engine, @id_company);",
    addengine_platform: "INSERT INTO game_engine_platform (id_engine, id_platform) VALUES(@id_engine, @id_platform);",


};
```

Funcionamiento de las APIS para la carga de datos:






BD2S22023_GRUPO_20
connection.js
db_creds.js
index.js
querys.js
node_modules
src
index.js
.gitignore
col.js
col2.js
collection.js
game_engine.js
game_localizations.js
game.js
gamess.js
juegos.mjs
language.js
motorjuego.js
package-lock.json
package.json
player_persepctive.js
README.md
region.js
release_date.js
theme.js
APIPY
credenciales.txt
main.py
main2.py
OUTLINE
TIMELINE
SONARLINT ISSUE LOCATIONS
gamess.js 3, M
PROYECTO1 > API > APIJS > gamess.js > peticion
import fetch from 'node-fetch';
import { getConnection, querys, sql } from "./db/index.js";
var num = 95220; Unexpected var, use let or const instead.
const pool = await getConnection();
async function peticion() { Refactor this function to reduce its Cognitive Complexity from 93 to the 15 allowed. [+32 locations]
try {
const response = await fetch(
"https://api.igdb.com/v4/games",
{
method: 'POST',
headers: {
'Accept': 'application/json',
'Client-ID': 'rf5rz2l7mhh48x5q4dyqyuiikr3dyg',
'Authorization': 'Bearer 1hkmbgavqo7ztv6wvc71gts84u4z1q',
},
PROBLEMS 3
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
CODEWHISPERER REFERENCE LOG
97012 agregado correctamente
97013 agregado correctamente
97014 agregado correctamente
97015 agregado correctamente
97016 agregado correctamente
97017 agregado correctamente
97018 agregado correctamente
97019 agregado correctamente
NUM: 97030
97020 agregado correctamente
97021 agregado correctamente
97022 agregado correctamente
97023 agregado correctamente
97024 agregado correctamente
97025 agregado correctamente
97026 agregado correctamente
node
node
carga/oscar*
AWS
Screen Reader Optimized
Ln 9, Col 10
Spaces: 4
UTF-8
CRLF
JavaScript
Prettier
10:28 PM
22-Sep-23


PROYECTO1
gamess.js
PROYECTO1
API
APIJS
db
node_modules
src
.gitignore
col.js
col2.js
collection.js
game_engine.js
game_localizations.js
game.js
gamess.js
juegos.mjs
language.js
motorjuego.js
package-lock.json
package.json
player_persepctive.js
README.md
region.js
release_date.js
theme.js
APIPY
OUTLINE
TIMELINE
API > APIJS > gamess.js > peticion
import fetch from 'node-fetch';
import { getConnection, querys, sql } from "./db/index.js";
var num = 150181;
const pool = await getConnection();
async function peticion() {
try {
const response = await fetch(
"https://api.igdb.com/v4/games",
{
method: 'POST',
headers: {
'Accept': 'application/json',
'Client-ID': 'etamy4nugfs5ujw3jaiond9wlw77j1',
'Authorization': 'Bearer rrnmuct08yldcqtj2p9y41y6u9tp9w',
},
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
node - APIJS
191554 agregado correctamente
191555 agregado correctamente
191556 agregado correctamente
191557 agregado correctamente
191558 agregado correctamente
191559 agregado correctamente
191560 agregado correctamente
NUM: 191571
191562 agregado correctamente
191563 agregado correctamente
Live Share
Ln 100, Col 29
Spaces: 4
UTF-8
CRLF
JavaScript
ENG
12:55 a. m.
23/09/2023