Test Plan Design Report

Test Project: Video Games Rest API
Test Plan: Test plan Rest API Video Games
Test Suite: Rest API Video Games

Printed by TestLink on 01/02/2023

Test Project: Video Games Rest API

To verify the Rest API for Video Games test application in swagger

Test Suite : Rest API Video Games

Test Case VG-1: GET video games		
Author:	cmendoza	
Summary: Returns all the videos games in the DB		
Preconditions: To have Video Game Database - Test Application running		
#:	Step actions:	Expected Results:
1	Create a session to VideoGamesDB	

2	Create a GET session to app/videogames	The application should display all games in JSON format like:
		
		VideoGame {
		id (integer, optional),
		name (string, optional),
		releaseDate (string, optional),
		reviewScore (integer, optional),
		category (string, optional),
		rating (string, optional)
		}
		
		The Response code should be 200

Execution type:	Manual	
Estimated exec. duration (min):		
Priority:	Medium	

Test Case VG-2: POST video games		
Author:	cmendoza	
Summary: Add a new video game to the DB		
Preconditions: To have Video Game Database - Test Application running		
#:	Step actions:	Expected Results:
1	Create a session to VideoGamesDB	
2	Create a POST session to app/videogames/ to add a new video game with those parameters: VideoGame {	The application should add a new video game to the DB The Response code should be 200

	id (integer, optional), name (string, optional), releaseDate (string, optional), reviewScore (integer, optional), category (string, optional), rating (string, optional) }	The response body should be : {"status": "Record Added Successfully"} If the parametres are incorrect the application display a error 400 If the item was already added the application display error 500 If the game id is invalid display the error 404
Execution type:	Manual	
Estimated exec. duration (min):		
Priority:	Medium	

Test Case VG-3: DELETE video games		
Author:	cmendoza	
Summary: Add a new video game to the DB		
Preconditions: To have Video Game Database - Test Application running		
#:	Step actions:	Expected Results:
1	Create a session to VideoGamesDB	
2	Create a DELETE session to app/videogames/{videoGameId} to delete a video game with those parameters: Parameter: VideoGameId Value example: 6 Data type: integer	The application should add a new video game to the DB The Response code should be 200 The response body should be : {"status": "Record Added Successfully"} If the parametres are incorrect the application display a error 400 If the item was already added the application display error 500 If the game id is invalid display the error 404
Execution type:	Manual	
Estimated exec. duration (min):		
Priority:	Medium	

Test Case VG-8: PUT video game by GameId		
Author:	cmendoza	
Summary: Returns a video game from the DB		
Preconditions: To have Video Game Database - Test Application running		
#:	Step actions:	Expected Results:
1	Create a session to VideoGamesDB	

2	Create a PUTsession to app/videogames/{videoGameId}
	
	Parameters: videoGameId
	Value: 1
	Data Type: integer
	
	Send the body parameters
	
	VideoGame {
	id (integer, optional),
	name (string, optional),
	releaseDate (string, optional),
	reviewScore (integer, optional),
	category (string, optional),
	rating (string, optional)
	}
		The request should return the {videoGameId}
		
		Response Body
		
		VideoGame {
		id (integer, optional),
		name (string, optional),
		releaseDate (string, optional),
		reviewScore (integer, optional),
		category (string, optional),
		rating (string, optional)
		}
		
		The Response code should be 200
		
		If the game id is not in the DB the application display a error 500
		
		If the game id is invalid display the error 404

Execution type:	Manual	
Estimated exec. duration (min):		
Priority:	Medium	

Test Case VG-6: GET video game by GameId		
Author:	cmendoza	
Summary: Returns a video game from the DB		
Preconditions: To have Video Game Database - Test Application running		
#:	Step actions:	Expected Results:
1	Create a session to VideoGamesDB	
2	Create a GET session to app/videogames/{videoGameId}	The request should return the {videoGameId} VideoGame {

	Parameters: videoGameId Value: 1 Data Type: integer	id (integer, optional), name (string, optional), releaseDate (string, optional), reviewScore (integer, optional), category (string, optional), rating (string, optional) } The Response code should be 200 If the game id is not in the DB the application display a error 500 If the game id is invalid display the error 404
Execution type:	Manual	
Estimated exec. duration (min):		
Priority:	Medium	