Givdo API Server

The Givdo Node API Server is hosted on Github here:

https://github.com/Givdo/givdo_node_api_server

It provides an internal database and API server that can be used to access database records via a REST API. To install and configure the server, follow the instructions in the readme. If you have a problem with installation, please speak to the server team for help.

Notes:

1. All data is returned in a JSON array. If the call completes without error but the specified data does not exist or has been inactivated, the server will return an empty array.

2. All required attributes must be passed in JSON format in the request body. Refer to the Postman collection for examples of each individual call.

3. HTTP DELETE methods are provided, however, for security reasons, nothing is actually deleted from the database via the API. (To allow the client to delete database records could lead to loss of important data.) Instead, when a DELETE method is invoked, records are merely marked as inactive in the database. To actually delete records that really should be deleted, you will need to use database tools (such as SQL queries or database UIs) with admin privileges.

The following table summarizes the API endpoints:

Item	HTTP Method	API End Point	Description
A1	GET	/users	Returns all active users
A2	GET	/users/{user_id}	Returns user with given id (if it exists and is active)
A3	POST	/users/insert	Adds user to database Required attributes: first_name, last_name Optional attributes: email, image_link, facebook_id
A4	PUT	/users/{user_id}	Updates user by id Required attributes: first_name, last_name Optional attributes: email, image_link, facebook_id
A5	DELETE	/users/{user_id}	Deactivates user by id
B1	GET	/badges	Returns all active badges
B2	GET	/badges/{badge_id}	Returns badge with given id (if badge exists and is active)
B3	POST	/badges/insert	Adds a new badge to the database Required attributes: name, image_link, score
B4	PUT	/badges/{badge_id}	Updates badge by id Required attributes: name, image_link, score
B5	DELETE	/badges/{badge_id}	Deactivates badge by id
C1	GET	/causes	Returns all active causes
C2	GET	/causes/{cause_id}	Returns cause with given id (if cause exists and is active)
C3	POST	/causes/insert	Adds a new cause to the database. Required attributes: name, image_link
C4	PUT	/causes/{cause_id}	Updates cause by id. Required attributes: name, image_link
C5	DELETE	/causes/{cause_id}	Deactivates cause by id
D1	GET	/questions	Returns all active questions
D2	GET	/questions/{question_id}	Returns question with given id (if it exists and is active)
D3	POST	/questions/insert	Adds question to database Required attributes: question_text, category_id
D4	PUT	/questions/{question_id}	Updates question by id Required attributes: question_text, category_id
D5	DELETE	/questions/{question_id}	Deactivates question by id
E1	GET	/question_options/{id}	Returns all active question options for the specified id
E2	GET	/question_options/{question_id}/{question_option_id}	Returns question option for a given question and question option id (if it exists and is active)
E3	POST	/question_options/insert	Adds question option to database Required attributes: text, question_id, is_correct
E4	PUT	/question_options/{question_option_id }	Updates question option by id Required attributes: text, question_id, is_correct
E5	DELETE	/question_options/{question_option_id}	Deactivates question option by id
F1	GET	/question_categories	Returns all active question categories
F2	GET	/question_categories/{question_category_id}	Returns question category with given id (if it exists and is active)
F3	POST	/question_categories/insert	Adds question category to database Required attribute: name
F4	PUT	/question_categories/{question_category_id }	Updates question category by id. Required attribute: name
F5	DELETE	/question_categories/{question_category_id}	Inactivates question category by id
G1	GET	/games	Returns all game records
G2	GET	/games/{creator_user_id}	Returns all games created by a specific user
G3	POST	/games/insert	Adds a game to the database Required attributes: creator_user_id, single_player
G4	PUT	/games/{game_id}	Updates game by id Required attributes: creator_user_id, single_player
G5	PUT	/games/finalize/{game_id}	Finalizes game (sets end time in database) by game id; call this when game is finished
H1	GET	/game_questions/{game_id}	Returns all game questions with given game id
H2	POST	/game_questions/insert	Adds a game question to the database Required attributes: game_id, question_id
I1	GET	/player_response	Returns all player responses
I2	GET	/player_response/{player_response_id}	Returns player response with given id (if exists)
I3	POST	/player_response/insert	Adds a player response to the database Required attributes: user_id, game_id, question_id, question_option_id
J1	GET	/donations	Returns all donations
J2	GET	/donations/{donation_id}	Returns donation with given id (if it exists)
J3	POST	/donations/insert	Adds a donation to the database Required attributes: user_id, item_id, organization_id, is_monetary, amount
J4	PUT	/donations/{donation_id}	Updates donation by id Required attributes: user_id, item_id, organization_id, is_monetary, amount
K1	GET	/donation_items	Returns all active donation items
K2	GET	/donation_items/{donation_item_id}	Returns donation item with given id (if it exists and is active)
K3	POST	/donation_items/insert	Adds donation item to database Required attributes: name, category, description
K4	PUT	/donation_items/{donation_item_id}	Updates donation item by id Required attributes: name, category, description
K5	DELETE	/donation_items/{donation_item_id}	Deactivates donation item by id
L1	GET	/user_badges/{user_id}	Returns all active user badges associated with a given user id
L2	POST	/user_badges/insert	Adds user badge to database Required attributes: user_id, badge_id
L3	DELETE	/user_badges/{user_id}/{badge_id}	Deactivates user badge by user id and badge id
M1	GET	/user_causes/{user_id}	Returns all active user causes associated with a given user id
M2	POST	/user_causes/insert	Adds user cause to database Required attributes: user_id, cause_id
M3	DELETE	/user_causes/{user_id}/{cause_id}	Deactivates user causes by user id and cause id
N1	GET	/user_game_attempts	Returns all user game attempts
N2	GET	/user_game_attempts/{user_id}	Returns all user game attempts associated with a given user
N3	GET	/user_game_attempts/{user_id}/{game_id}	Returns user game attempt with given user id and game id
N4	POST	/user_game_attempts/insert	Adds a user game attempt to the database Required attributes: user_id, game_id, score, won
N5	PUT	/user_game_attempts/{game_id}	Updates user game attempt by id Required attributes: user_id, game_id, score, won
O1	GET	/admin_users	Returns all active admin users
O2	GET	/admin_users/{admin_user_id}	Returns admin user with given id (if it exists and is active)
O3	POST	/admin_users/insert	Adds admin user to database Required attributes: first_name, last_name, email, encrypted_password
O4	PUT	/admin_users/{admin_user_id}	Updates admin user by id Required attributes: first_name, last_name, email, encrypted_password
O5	DELETE	/admin_users/{admin_user_id}	Deactivates admin user by id
P1	GET	/organizations	Returns all active organizations
P2	GET	/organizations/{organization_id}	Returns organization with given id (if organization exists and is active)
P3	POST	/organizations/insert	Adds a new organization to the database. Required attributes: name, street address, city, state, zip, mission Optional attributes: facebook_id, image_link
P4	PUT	/organizations/{organization_id}	Updates organization by id Required attributes: name, street address, city, state, zip, mission Optional attributes: facebook_id, image_link
P5	DELETE	/organizations/{organization_id}	Deactivates organization by id
Q1	GET	/advertisements	Returns all active advertisements
Q2	GET	/advertisements/{advertisement_id}	Returns advertisement with given id (if it exists and is active)
Q3	POST	/advertisements/insert	Adds advertisement to database Required attributes: company_name Optional attributes: image_link
Q4	PUT	/advertisements/{advertisement_id}	Updates advertisement by id Required attributes: company_name Optional attributes: image_link
Q5	DELETE	/advertisements/{advertisement_id}	Deactivates advertisement by id
R1	POST	/auth/facebook/callback	Logs a user in with Facebook. The client must have obtained authorization from the user and received a Facebook access token. The client needs to pass the Facebook access token to this call in JSON format in the request body. Required attributes: access_token See Givdo Auth Demo and code comments in config/passport.js for more information.
R2	GET	/checklogin	Checks to see if a user is currently logged in. Requires a JSON web token passed in the headers with label: Authorization, and id: JWT. Required attributes: JWT in Authorization header See Givdo Auth Demo and code comments in config/passport.js for more information.