RPG_Project API

API Calls

API calls are generally formatted as follows:

hostURL/<router>/<function>/<tableName>/<ID>

Router
- The ExpressJS router that the API call is being sent to. At the time of writing this document, the only route that has been implemented is */test*. More routes will be added as the front end continues to be developed.

Function
- The action being performed. This determines what SQL will be generated for the query. Examples include *view, edit, delete,* ext…

Table Name
- The database table that the action is being performed on.

ID
- The ID of the entry that the action is being performed on. Only included if the action is being performed on a particular entry in the table.

Payloads

Data from the database is retrieved as JSON in the following format:

```
{
        0: {
                Field: entry,
                Field: entry
        }
        1: {
                Field: entry,
                Field: entry
        }
        ....
}
```

Each row is indexed in the response JSON by an integer, starting at 0. This is true even if the request only asks for a single entry.

To access a particular entry in the front end code, the following format must be used:

```
const <xhttp request variable> = new XMLHttpRequest();

<xhttp request variable>.open('<method>','<API route>');
<xhttp request variable>.onload = function () {

        // The code inside the onload function will not run until the database has sent the response to the front end script
        let <data variable> = JSON.parse(<xhttp request variable>.responseText);
        let <entry variable> = <data variable>[<row index>][<field name>];

        // Do things with the response data here
}
<xhttp request variable>.send();
```

There may be times where data needs to be read from the URL on the front end script *(usually the ID of a row that is going to be edited in some way).* In such cases, the data from the last section of the URL can be retrieved by importing **parseData** from **'./parseURL.js'** and storing the return value of **parseData(location.href);** as a variable.

ParseData() is designed to interpret the last section of the URL *(formatted as '<field name>':<value>-'<field name>':<value>)* as JSON by replacing the single quotes with double quotes, delineating entries with '-', and adding opening and closing brackets. The function then returns a JavaScript object with all of the data in it.

(Note: In the URL, field names and string values will need to be surrounded by single quotes, non string values should not be surrounded by anything)