API research

By Ruan Klopper

Student no. 231280

Subject: DV200

Date: 15/02/2024

Ideas for API: Cellphone specs, car specs

Open Source APIs: https://github.com/public-apis/public-apis?tab=readme-ov-file#vehicle

List of possible API

1. **GSMArena: https://github.com/nordmarin/gsmarena-api?tab=readme-ov-file#installation**
2. **TechSpecs API: https://developer.techspecs.io/pricing/**
3. **Comics: https://comicvine.gamespot.com/api/documentation**
4. **Meals: https://www.themealdb.com/api.php**
5. **Recipe: https://rapidapi.com/spoonacular/api/recipe-food-nutrition**
6. **SpaceX: https://docs.spacexdata.com/**

API 1: SpaceX launches API

API Description: The SpaceX Launches API provides information about launches conducted by SpaceX, including details such as launch dates, mission names, rocket types, payloads, and more.

Link to documentation: https://docs.spacexdata.com/

API call example

CALL TYPE: GET Ajax

```
$.ajax({
  url: "https://api.spacexdata.com/v4/launches",
  dataType: "json",
  success: function(data) {
    // Process the retrieved launch data here
    console.log("SpaceX launches:", data); // Example: Log all
launches to console

    // You can display the data on your webpage using DOM
manipulation, e.g.:
    const launchList = $("#launch-list"); // Assuming you have a list
element

    data.forEach(function(launch) {
      const listItem = $("<li></li>");
      listItem.text(`${launch.mission_name} -
${launch.launch_date_local}`);
      launchList.append(listItem);
    });
  },
  error: function(jqXHR, textStatus, errorThrown) {
    console.error("Error retrieving launch data:", textStatus,
errorThrown);
  }
});
```

DATA OUTPUT: JSON:

```
Properties:
   •  fairings
         •  reused
         •  recovery_attempt
         •  recovered
         •  ships
   •  links
         •  patch
               •  small
               •  large
         •  reddit
```

```
            • campaign
            • launch
            • media
            • recovery
      • flickr
            • small
            • original
      • presskit
      • webcast
      • youtube_id
      • article
      • wikipedia
• static_fire_date_utc
• static_fire_date_unix
• net
• window
• rocket
• success
• failures
      • time
      • altitude
      • reason
• details
• crew
• ships
• capsules
• payloads
• launchpad
• flight_number
• name
• date_utc
• date_unix
• date_local
• date_precision
• upcoming
• cores
      • core
      • flight
      • gridfins
      • legs
      • reused
      • landing_attempt
      • landing_success
      • landing_type
      • landpad
• auto_update
• tbd
• launch_library_id
• id
```

- **Outline of comparative data:**
  1. **Mission Name**
  2. **Launch Date**

    3. **Rocket Type**
    4. **Payload Information**
    5. **Launch Success Status**
- **Outline of timeline data:**
    1. **Mission Name**
    2. **Launch Date**
    3. **Rocket Type**
    4. **Failure Details**
    5. **Rocket Recovery Details**

API 2: Comic Vine

API Description:

Link to documentation: https://comicvine.gamespot.com/api/

API call example

URLs

Get details by name:
https://comicvine.gamespot.com/api/search/?api_key=${apiKey}&format=json&resources=issue&query=${comicTitle}

Get details by issue ID:
https://comicvine.gamespot.com/api/issue/<issue_id>?api_key=<your_api_key>&format=json

CALL TYPE: GET - Get Comic book details by name

```
function getComicDetails(comicTitle) {
  // Replace YOUR_API_KEY with your actual Comic Vine API key
  const apiKey = "YOUR_API_KEY";

  // Build the API endpoint URL
  const url =
`https://comicvine.gamespot.com/api/search/?api_key=${apiKey}&forma
t=json&resources=issue&query=${comicTitle}`;
  $.ajax({
    url,
    dataType: "json",
    success: function(data) {
      // Check if any results were found
      if (data.results.length > 0) {
        const firstResult = data.results[0];

        // Extract relevant details from the first result
        const issueName = firstResult.name;
        const volumeName = firstResult.volume.name;
        const coverImage = firstResult.image.small_url; // Replace
"small_url" with desired image size

        // Display the details in your desired format (e.g., using
jQuery)
        $('#comic-details').html(`
          <h2>${issueName}</h2>
          <p>Volume: ${volumeName}</p>
          <img src="${coverImage}" alt="${issueName} cover">
        `);
      } else {
        // Handle the case where no comic was found
        console.error("No comic found with title:", comicTitle);
```

```
          }
        },
        error: function(jqXHR, textStatus, errorThrown) {
          console.error("Error:", textStatus, errorThrown);
        }
      });
    }
```

DATA OUTPUT: JSON:

```
1. error
2. limit
3. offset
4. number_of_page_results
5. number_of_total_results
6. status_code
7. results
   - aliases
   - api_detail_url
   - cover_date
   - date_added
   - date_last_updated
   - deck
   - description
   - has_staff_review
   - id
   - image
      - icon_url
      - medium_url
      - screen_url
      - screen_large_url
      - small_url
      - super_url
      - thumb_url
      - tiny_url
      - original_url
      - image_tags
   - associated_images
   - issue_number
   - name
   - site_detail_url
   - store_date
   - volume
      - api_detail_url
      - id
      - name
      - site_detail_url
   - resource_type
8. version
```

- **Outline of comparative data:**
  1. **Date added**
  2. **Review tbc**
  3. **Characters – Must do another api call to obtain characters**
- **Outline of timeline data:**
  1. **Name**
  2. **Description**
  3. **Date added**
  4. **Deck**
  5. **review**

Favorite

API 3: Recipe - Food - Nutrition

API Retrieved from RapidAPI: https://rapidapi.com/

API Description:

The spoonacular Recipe - Food - Nutrition API gives you to access to thousands of recipes, storebought packaged foods, and chain restaurant menu items. Our food ontology and semantic recipe search engine makes it possible to search for recipes using natural language queries, such as "gluten free brownies without sugar" or "low fat vegan cupcakes." You can automatically calculate the nutritional information for any recipe, estimate recipe costs, visualize ingredient lists, find recipes for what's in your fridge, find recipes based on special diets, nutritional requirements, or favorite ingredients, classify recipes into types and cuisines, convert ingredient amounts, or even compute an entire meal plan. With our powerful API, you can create many kinds of food and nutrition apps.

Special diets/dietary requirements currently available include: vegan, vegetarian, pescetarian, gluten free, grain free, dairy free, high protein, low sodium, low carb, Paleo, Primal, ketogenic, and more.

Author: https://rapidapi.com/user/spoonacular

Link to documentation: https://rapidapi.com/spoonacular/api/recipe-food-nutrition

API call example

CALL TYPE: GET – Search recipes

```
const settings = {
async: true,
crossDomain: true,
url: 'https://spoonacular-recipe-food-nutrition-
v1.p.rapidapi.com/recipes/search?query=burger&diet=vegetarian&excludeI
ngredients=coconut&intolerances=egg%2C%20gluten&number=10&offset=0&typ
e=main%20course',
method: 'GET',
headers: {
'X-RapidAPI-Key':
'7a92345447mshd0df87618117100p1469ddjsn05b06500b351',
'X-RapidAPI-Host': 'spoonacular-recipe-food-nutrition-
v1.p.rapidapi.com'
}
};

$.ajax(settings).done(function (response) {
console.log(response);
});
```

DATA OUTPUT: JSON:

```
Output structure of Search Recipies by Name:

   1. Properties:
         •  results (array)
  1. id (number)
  2. title (string)
  3. readyInMinutes (number)
  4. image (string)
  5. imageUrls (array)
         •  baseUri (string)
         •  offset (number)
         •  number (number)
         •  totalResults (number)
         •  processingTimeMs (number)
         •  expires (number)
         •  isStale (boolean)
   2. Subproperties within the results array:
         •  imageUrls (array) (inside each item)


Output structure of Get Recipe Information by ID

   1. Properties:
         •  vegetarian (boolean)
         •  vegan (boolean)
         •  glutenFree (boolean)
         •  dairyFree (boolean)
         •  veryHealthy (boolean)
         •  cheap (boolean)
         •  veryPopular (boolean)
         •  sustainable (boolean)
         •  weightWatcherSmartPoints (number)
         •  gaps (string)
         •  lowFodmap (boolean)
         •  ketogenic (boolean)
         •  whole30 (boolean)
         •  servings (number)
         •  sourceUrl (string)
         •  spoonacularSourceUrl (string)
         •  aggregateLikes (number)
         •  creditText (string)
         •  sourceName (string)
         •  extendedIngredients (array)
         •  id (number)
         •  title (string)
         •  readyInMinutes (number)
         •  image (string)
         •  imageType (string)
         •  instructions (string)
   2. Subproperties within the extendedIngredients array:
         •  id (number)
         •  aisle (string)
```

- `image (string)`
- `name (string)`
- `amount (number)`
- `unit (string)`
- `unitShort (string)`
- `unitLong (string)`
- `originalString (string)`
- `metaInformation (array)`

- **Outline of comparative data:**
  1. **weightWatcherSmartPoints**
  2. **servings**
  3. **aggregateLikes**
  4. **will also use the "Nutrition by ID" which will provide additional information on nutrition that will help with creating charts**
     <u>outline of nutritious:</u>

ingredients (array)

- **amount (object)**
  - **metric (object)**
    - **unit (string)**
    - **value (number)**
  - **US (object)**
    - **unit (string)**
    - **value (number)**
- **image (string)**
- **name (string)** ***Everyone is unique***

- **Outline of timeline data:**
  1. **Ingredients**
  2. **Popular**
  3. **Cheap**
  4. **veryHealty**

PROS:

1. **Containing everything I need**
2. **Does have a lot of data to work with**

CONS:

1. **500 requests per day then USD 0.007 there after**