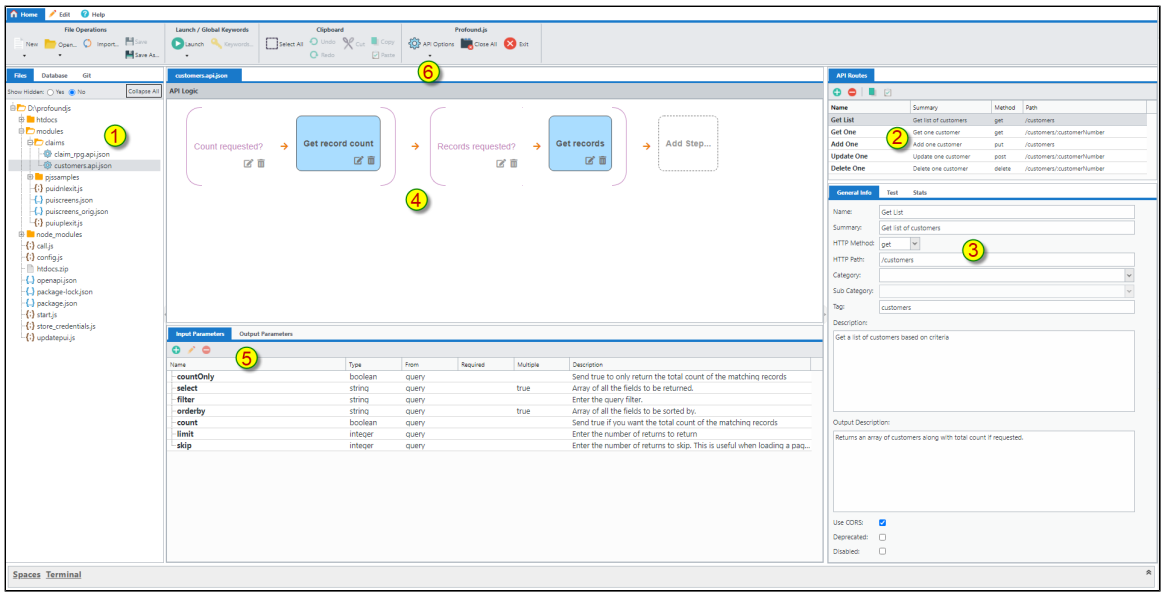
API Designer

To open the API designer, open the Profound.js IDE and click **New > API File** to add a new API, or edit an existing file with the extension "api.json."

While you are editing or adding an API File, the following API Designer will be presented:



This API Designer has several sections:

1. **Files Tree** tab: In this example, you'll see that an API file (a filename that ends with .api.json) was opened.

2. **API Routes** section: This is a list of APIs stored within this file. Here is where you will select, add, update, copy, and remove APIs.

3. This is where you will provide information about the selected API.

- Within the **General Info** section:
  - **Name**: This is used internally and needs to be unique to this file.
  - **Summary**: This is a short description of what this API does.
  - **HTTP Method**: The method type used to access this API: 'get' (retrieve information), 'delete' (delete information), 'put' (update information), and 'post' (everything else)
  - **HTTP Path**: This is the URL path that will activate this API. For example:
  - `/customers` This would represent a RESTful API to get a list of customers
  - `/customer/:id` This would represent a RESTful API to get a single customer for which its ID column equals this path input parameter
  - **Category** and **Sub Category**: These allow you to categorize APIs for easier search and maintenance. These fields are used to create a directory of categorized APIs when you use the Find API File option.
  - **Tag**: Another way to categorize your APIs.
  - **Description**: Describes this API. This field supports CommonMark. This will help the consumers of this API as well as future maintainers of this API.
  - **Output Description**: Describes what is being returned by this API. This field supports CommonMark.
  - **Use CORS**: Check this if you want this API to be accessible to other websites, including other internal web servers.
  - **Deprecated**: Check this if you want to mark this API as deprecated or soon to be removed. The API will appear as deprecated in the API Explorer.
  - **Disabled**: Check this if you want to disable this API from being served.
- In the **Test** section:
  - You can test your API, seeing all of the parameters along with your API result
- In the **Stats** section
  - You can view usage about this API, such as
    - Number of requests
    - Average response time
    - Average payload size
    - And Much more

4. **API Logic** section: Use Low-Code steps and plugins to program the selected API routine.

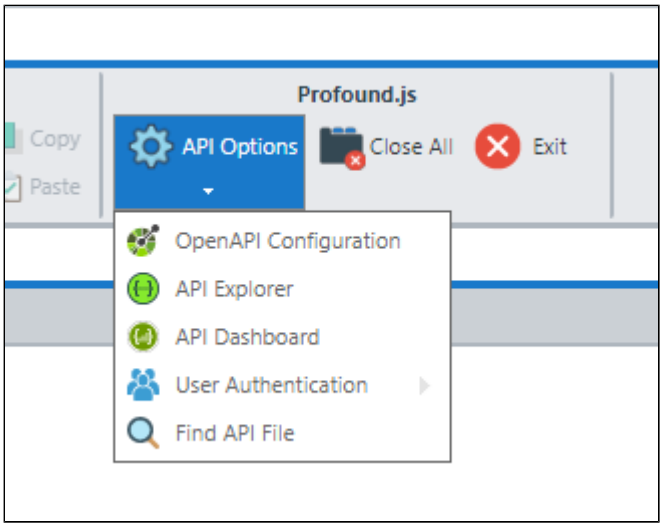
5. **Input Parameters** and **Output Parameters**: This is where you will define inputs and outputs for the selected API, as well as order and document them.

- All parameters are validated before running any of the API logic.
- You can add nested child parameters. Simply add a parameter with the type "object", select it, and click the add parameter icon again. The new parameter will be added as a child.
- You can also drag-and-drop the parameters to order them as you like.

For each parameter you define, you can specify:

- **Name**: Unique name for this parameter (at this level)
- **From**: Where will the input parameter come from
    - query is for simple values and used with GET HTTP methods
    - header parameter are not commonly used as parameters, but they are supported
    - body is for more complex values such as all of the parameters for adding a new customer
        - body parameters are not supported with HTTP GET or DELETE methods.
- **Data type**: The type of value that will be sent to the API.
    - On input parameters and when "from" is set to "body", a new item becomes available called "object", allowing for complex parameters.
    - If a parameter is defined as an integer and the parameter value contains alpha characters, an error will be returned to the caller before your API logic.
    - Same is true for all parameters. All values sent must be convertible to this type. If not an error is returned to the caller before your API logic.
- **Required**: Check if the input parameter must be passed and must have a valid value.
    - When a parameter is defined as required and the parameter is not passed, an error will be returned to the caller before your API logic is called.
- **Example**: An example of what the parameter value should be. This is an example value not a default value.
- **Allow Multiples**: Check if the input parameter should be an Array.
    - If a input parameter is defined as an Array, but parameter value passed is not an array, an array will be created with that value as the only element, and then passed to your API logic
    - If a input parameter is not defined as an Array, but parameter value is an array, the first valid value from that array will be passed to your API logic
- **Is Array**: Check if the output parameter will be returned as an Array.
- **Deprecated**: Check this if you want to mark this parameter as deprecated or soon to be removed. The parameter will appear as deprecated in the API Explorer.
- **Description**: Describes the parameter. This field supports CommonMark.

6. **API Options** section: This is where you can access other API tools from the IDE.



- **OpenAPI Configuration**: This will open the openapi.json file where you can set up:
    - Your public API information, such as title, contact info, version, and license info-object
    - (Coming soon) API Security configuration security-scheme-object
- **API Explorer** - This will open a new browser tab where you can explore all of the enabled APIs
- **API Dashboard** - This will open a new browser tab where you can see more statistics about the server and all the enabled APIs
- **User Authentication:**
    - (Coming soon) Administrator credentials for API usage
    - Administrator credentials for API Dashboard access
- **Find API File** - If you do not know the API filename, then this is a great way to find an API