Echo Service Documentation

# Release notes // V1.0

This is a service that provides mock API data based on specified templates. The current version has the following features

* Browse through and sort a list of all available templates
* Create a new template by uploading a json file and filling out the metadata.
* View the template and preview the response.
* Edit template keys, values and value types.
* Create and browse through multiple versions of the same template. For example, one has static data that’s always the same, and the other uses dynamic data and shows a random email each time you call it.
* Ability to set “required” keys that must be included using POST when accessing the template.
* An “echo” service that responds with whatever the post body was.

# Project Structure

**/src** contains all the necessary files required for the service to function. Only the files in the *public* folder should be accessible to the user, all the rest are “behind the curtain” files that are needed but should not be visible.   
   
Within you will find:

**/css** has a style.css and a style.map.css which are files generated by the SCSS compiler.

**/db** contains the two sqlite database files the project uses.

* **echo.sqlite** contains all the templates and related information
* **random.sqlite** contains various values such as emails and names that are used to populate dynamic values.

**/dependencies** contain 3rd party libraries used in this project

* **/boostrap** has been used for styling, as it’s UI is a common and familiar sight. Bootstrap makes it clear what’s a button and what’s just information.
* **/datatables** is a library used to make html tables dynamic and sortable. This is used on the homepage (*../public/index.php*) to make the list of templates interactive.
* **/fontawesome** has been used for all of the icons on the website.
* **/jquery** contains the minified v3.5.1 of jQuery which is both – required by some of the other dependencies and I myself use throughout the website.

**/img** has the favicon. Right now it’s a placeholder with the logo of the police

**/js** has all the custom javascript I wrote for the website

* **branch.****js** is present on *../public/edit.php* and handles the “create branch” functionality. It gets all of the template information and sends it using ajax to *../php/branch.php* that uses the post data to create a new branch in *echo.sqlite*. It uses functions and is dependent on *json.js*.
* **clear.js** is only present on the *../public/new.php*page and handles the “clear” button, which is remove the uploaded template and any entered inputs like text.
* **collapse.js** handles showing and hiding nested structures within the template.
* **datatables.js** is present on ../public/index.php and initializes the library.
* **delete.js** is responsible for all of the delete functionality in the app. It handles removing individual versions as well as removing a template completely.
* **json.js** is the most important file, and is what converts the html table to a json format. It has the functions to redraw the preview, and it’s what branch, update and create.js use to get the template data that they sent.
* **update.js** is present on *../public/edit.php* and handles the “update version” functionality. Just like branch.js, it gets all template data and sents it using ajax to ../php/update.php

**/php** has the “background” php files that handle creating, updating or deleting records from the echo.sqlite database.

* **branch.php** handles creating a new version of a template (a new branch)
* **create.php** handles creating a completely new template
* **delete.php** handles all of the deletions
* **update.php** handles updating an existing version of a template.

**/public** contains the actual pages the user will access

* **index.php** is the homepage. It shows a list of all currently available templates and links to them, has a button to create a new one, and allows filtering the list to find the template that you need
* **new.php** is the page on which you create a new template.
* **view.php** is the view (and preview) of the template. Here you can switch between and view individual versions.
* **edit.php** is where the user will go to edit a template. or create a new branch.
* **echo.php** is the endpoint of the api, from where you can GET or POST specified templates.

**/scss** contains the the actual style and bootstrap overrides.

* **\_components.json.scss** contains the styles needed for syntax highlighting in the preview window, with the rest following the regular scss/sass conventions

# Installation Guide & Requirements

## Installation

Put the contents of the **/src** folder in the root directory of your server, and start it up.   
Open a recent browser like Firefox or Chrome and navigate to <server>/public to start using the service.

## Requirements:

* PHP 7.4.2
* ECMAScript 6
* HTML5

## Database structure

A screenshot of a cell phone

Description automatically generated<https://dbdiagram.io/d/5ef15b649ea313663b3aeb3c>

template\_info contains the “general” information used to populate the list on the index.php page.

template\_data contains the actual versions and its primary key is what’s used to specify the template when using echo.php

department has all the different departments, and templates\_departments is the in-beteen table that contains the information about which templates belong to what departments.

# User Guide

## Homepage // index.php

A screenshot of a cell phone

Description automatically generated

This is the homepage. It has a filter so you can browse the list by template name or author name. You can set the amount of rows you want to display per page or click the button to create a new template.

Clicking on the name of a template (2nd column) will take you to view.php where you can see the template’s details.

## Create template // new.php

A screenshot of a social media post

Description automatically generated

To create a template you must first upload the template.json file. The service will process it and display it and a preview in the corresponding windows.

If needed, you can then edit the keys, values and their types. If you need to change the structure however, such as adding or removing rows, you will need to use the clear button and reupload a new file.

A screenshot of a social media post

Description automatically generatedA screenshot of a social media post

Description automatically generated

After you are sure the template is correct you can fill out the metadata and press the create button. This will create the needed records in the SQLite database and take you to the view.php page to show you the template.

A screenshot of a cell phone

Description automatically generated

## View template // view.php

A screenshot of a social media post

Description automatically generated

You can view an existing template by selecting it from the list on the homepage. You are shown the template’s structure as well as the ULR to access it as an API with a preview of the response on the right.

From here you can use the dropdown to browse and view other versions based on this template, edit it, or delete the currently version.

A screenshot of a cell phone

Description automatically generatedCertain keys can be set as required, which means that the value is a placeholder, and you will need to overwrite it when accessing it through the API endpoint using a POST request with the required keys and value pairs.

Using the [[value]] syntax you can also set dynamic variables that will provide random data when used through the API

A screenshot of a cell phone

Description automatically generated

In this example, you will need to POST {“token”:”value”} and the “fdass” key will have a random email each time you call it.

## Edit template // edit.php

A screenshot of a social media post

Description automatically generated

It’s fairly self-explanatory, but you can the metadata of the template as well as its edit keys as well as their values and types.   
Editing the template structure is currently unsupported, and you will need to create a completely new template by uploading a new json file with changed structure to do it.

You can branch off to create a new version, or update the current one.

## Echo.php // the api endpoint

No user interface, see POSTMAN examples.

<https://www.getpostman.com/collections/cd2f0d1760819c0545c3>

This is the api endpoint where you can get the actual responses that you would use for testing your application.

If you do not specify any template using get, you can use POST to get an echo of its contents.

If the template has no required keys, it can simply be accessed using a get, when specifying it’s primary key. Otherwise, you will need to POST the required key value pairs using raw/JSON in the { “key”:”value”,”key2”:”value2”,…} format.

# Known bugs & issues

* If you delete the last “version” of a template, it will still show up in the template list, but the view.php will simply not load. You can delete that part from the homepage.   
  Proposed fix: check how many versions exist, and if there’s only one perform a complete delete instead of the partial one.
* If you fill out the metadata prior to uploading a file, it will be lost once it’s uploaded and you will have to re-enter it.  
  Proposed fix: submit the entered metadata together with the file and process it into the corresponding fields.
* Nested arrays cause the json generator in json.js to fail. I have not been able to figure out why. The php processes and displays them fine, but you won’t be able to save them. A temporary workaround is manually editing the template field in the SQLite database using SQLite Studio, but, again, you won’t be able to make use of the edit functionality until this issue is fixed