# Requirements definition

## Received description

1. Echo web service
2. Response emulation web service
3. Template based request frontend UI
4. Template based response frontend UI
5. Optional request structure validation
6. Optional response structure validation

## Interpretation

Create a REST API that will be used for testing requests of internal APIs and applications. The API will use PHP. Based on the endpoint and the type and data within the request, the echo api will respond with different things, such as:

1. Echo web service | POST  
   On receiving a POST request, the body of that request will be returned in the response, no matter the content type. It should mirror the body completely, and not add any extra messages or information.

* 1:1 mirroring
* Supported content types
  + - Form-data
    - X-www-form-urlencoded
    - Raw
    - Binary

Perhaps a nice to have would be a version of the api that could stream back the data in either purely binary (0 and 1) or base64, but I’m not sure this is necessary, need client feedback.

1. Response emulation web   
   In a similar being to echo, but the response is based on a template that is selected by the request. Request can contain a copy or part of the template, the values of which will be used in the response. The requests and templates will be in json or xml.
   * Request should be checked for amended template, and if there are corresponding keys, values from the request should be used in the response.  
     Keys will only be replaced when both their name and position are the same.
   * A mode that when selected will only returns the template, no status codes or messages
   * In addition to sending a request and getting back a template-based response, users should be able to upload, and get a list of all templates.
2. Template based request frontend UI (Create a custom request for #2)   
   This is a front-end web application in which you can load the templates used in #2, and create a custom request by changing the values of the keys in the template
   * Research if there are existing projects or tools that do something similar, I shouldn’t try to reinvent the wheel both structure and ui/ux wise. I need to look at how other form-generators do it and take the best parts.
     + Look into schemas
     + json needs to contain enough information to generate a proper html5 form + have some custom data, so per input we at minimum need:
       1. name (for label, id and name-tag) (this can be the key name)
       2. type (text, radio, submit, file, …)
       3. whether it’s required or not in the template
       4. values if it’s a select tag
   * Use a UI framework such as bootstrap for easier styling, it also makes for easier expandability, and given it’s prevalence on the web will be familiar and comfortable to the users
3. Template based response frontend UI (Create and change templates used in #2)  
   This is an extension of #3, but in addition to being able to change the values of a loaded template to generate a request, you could add or remove fields, change their names, types an whether they’re required, change the nesting structure etc.
   * You should also be able to make a template from scratch
   * You should be able to completely change every part of an existing template
   * Is some sort of version control necessary? Perhaps there could be a way to integrate git, so you could track and even select which version of the template you want to load.
     + Research if it’s possible to integrate git or some alternatives on server level, so it would automatically create a new branch per template name, and keep versions of them, merge them all into a develop or master etc…
4. Optional request structure validation (Extension of #3’s functionality)  
   This is an optional toggle than when used in #3 will validate the created custom request on whether it’s structure is valid, as well as compare it against the used template, for example check that no required keys are missing, that the structure is the same and so on. If necessary, it could even show a side-by-side comparison between the two, highlighting the differences.
   * How strict does it have to be?
     + Right now, if a key isn’t mentioned in the request, the response will contain the value of the template. Should a key that’s tagged as required always be present in the request?
     + What should the behavior be if the request has a key that doesn’t exist in the template? Right now it’s simply ignored, but if the structure validation is toggled it should cause the validation to fail?
     + If the data-type between a key in the request and template is different, it should cause the validation to fail.
5. Optional response structure validation (Extension of #4’s functionality)  
   An optional toggle that when used in #4 will check the if the code is valid json and validate the amended template to the previous version, to make sure the required keys are still there, and the data-types match.  
   This then should only be used when you expand upon a template, not change it.
   * + What languages need to be supported? Only json?