API Specification

In the target specification it is defined that there will be a web application. A web application consists of a front-end and a back-end module. The front – end is responsible for the user view and the back-end is responsible for the data handling. Both modules are linked through an API. If the API is defined first there are several advantages. The front-end developers doesn’t need to wait until the back-end developers finished implementing, they simply can rely on the agreement (the API) definition. The second advantage is that an API that is defined in first place is usually better thought out than an API that arouse while implementing the back-end module.

There are several ways to define an API we have agreed to use the RAML (ReST API Modelling Language) to define our API in first place. We want to use a resource oriented architecture and we therefore define a ReST-API.

# Reading the API with RAML

## Resources

Resources are introduced by a “/” followed by the resource name. Nesting is realized by indentation. Same indentation means that the resources are on the same level.

/user

/categories

The service can extract information out of the URL. URL parameters are values between /… / within in the URL. In RAML placeholder for those are defined with curly braces: /{userid}

Query parameters are another way of retrieving information from the URL. Query parameters are key-value pairs that are appended to the URL like [www.example.com/resource?key=value&key2=value](http://www.example.com/resource?key=value&key2=value).  
Those are specified within RAML as follows:

queryParameters:

page: integer

userid: string

The key value pairs are specified as seen with key : type.

## Methods on resources

After a resource specification the type of the http method is defined with method type and double colon. Everything that is intended below the method belongs to the method.

## Requests

The request body is specified by “body:” Afterwards the content-type follows, which is in our case application/json and then it is possible to define a json scheme with either “type:” or “scheme:” The scheme can be defined in place or in a separate file, referenced earlier in the RAML document.

***Example:***

post:

description: |

Create a new item

body:

application/json:

type: item

## Responses

Responses are defined in similar to the expected requests. The only difference is that the http response code comes first. Multiple responses with multiple http response codes are possible:

200:

description: |

Succesfull retrival of category list

body:

application/json:

schema: categorylist

404:

description: |

There are no categories to show