# LeBrisouBackend API FRONT END DEVELOPMENT

## [1] Description & Environment

This project is an entirely back-end tool, a database administration page. The main idea you will have to deal with is adding, searching and setting up a viewing and editing page for Json data coming from GET/POST query on the API. There are two servers :

- http://<domain>:8080/ is the Web Server
- http://<domain>:8081/ is the API Server

*note: for development facilities the back-end currently hold both API and Web Servers.*

I already put some time in Web development with Angular.Js, full public directory structure is normally setup. I usually work in local, on localhost, just checkout the project at this URL, LeBrisouBackend Repository and run “npm install”.

For testing purpose, there is a default file “*config/auth.json*” available, checkout it manually (since this file is in the “*.gitignore*”). Credentials are, user : test, password : test.

Then check the “package.json” for personal launch scripts. (running npm script others then start, test and so on can be done using “run-script” command : “npm run-script <script name>.

My advice is to constantly run : “npm run-script start dev”.

If you need a sample dataset, run the script *“script-test.sh”* in the *“server”* directory.

To prevent server from synchronizing and dropping the  the database model, turn *“db_option.sync.force” and “db_option.drop”* to false.

After that you'll be able to work on the front-end part without doing anything more.

## [2] Important informations

- Every Response from API Server contains at least :
    - “success” boolean field, meaningful ? :P
- In case of error you have :

```
{
    statusCode: 20000
    error: true
    message: “Sample Message”
    source: “LeBrisou-Backend”
}
```

- *Error fields are (some of them never appear in the response to the front):*

| StatusCode | Message |
|---|---|
| 20000 | Unknown Word |
| 20001 | Unknown Language |
| 20002 | Unknown Definition |
| 20003 | Unknown Country |
| 20004 | Unknown Synonym |
| 20005 | Unknown Antonym |
| 20006 | Unknown Example |
| 20007 | Unknown Relative |
| 20008 | Unknown Hyperlink |
| 20050 | Conflict: SynonymId equal WordId |
| 20051 | Conflict: AntonymId equal WordId |
| 20052 | Conflict : Duplicate Word : lema - pos |
| 20053 | Conflict: Duplicate Definition |
| 20054 | Conflict: Duplicate Definition - Example |
| 19 | Database Conflict |
| 21001 | Commit Failed |
| 21002 | Database is Empty |
| 19999 | Censored Error |
| 22000 | Reading Database init File |

## [3] Tasks

### [3.1] Add Query

- Method: POST
- URL: http://<domain>:8081/add
- parameters: Json formated Array containing data below

```
'lema' : Joi.string().required().example('desayuno'),
  'pos' : Joi.string().required(),
  'gerund': Joi.string().optional(),
  'participle': Joi.string().optional(),
  'countries': Joi.array().includes(
    Joi.object().keys({
      'country' : Joi.string().min(1).max(20).required(),
      'frequency': Joi.number().min(0).max(100).integer().optional()
    }).required()
  ).required(),
  'register': Joi.boolean(),
  'language': Joi.string().required(),
  'definitions': Joi.array().includes(
    Joi.object().keys(
      {
        'definition': Joi.string(),
        'examples': Joi.array().includes(Joi.string()).optional()
      }
    ).required()
  ),
  'synonyms': Joi.array().includes(
    Joi.object().keys(
      {
        'lema': Joi.string().required(),
        'pos': Joi.string().required()
      }
    ).required()
  ).optional(),
  'antonyms': Joi.array().includes(
    Joi.object().keys(
      {
        'lema': Joi.string().required(),
        'pos': Joi.string().required()
      }
    ).required()
  ).optional(),
  'relatives': Joi.array().includes(
    Joi.object().keys(
      {
        'lema': Joi.string().required(),
        'pos': Joi.string().required()
      }
    ).required()
  ).optional(),
  'hyperlinks': Joi.array().includes(Joi.string().required()).optional()
```

*note: You'll notice the response returned by the API Server is very verbose, keep in mind that this specification above is the only one you will use either for the forms or the query itself.*

- Page Specifications:
  - The front "Add" page design has to contain a html INPUT for each field described above (both optional and required).
  - Each INPUT can has either have "placeholder" or a text before specifying the field name. ("placeholder" is preferred).
  - "Synonyms", "Antonyms", "Hyperlinks" forms must be able to be added at will by the user in order to add more of them.
  - The same for "Definitions" and "Examples" related with the definition. (one definition can have many examples).
  - "register" is a check-box.
  - "hyperlinks" are valid URLs.
  - "frequency" is a number from 0 to 100.
  - "country" is a string from 3 to 32 chars.
  - "Add" button have to appear both at the top and the bottom of all forms
- Form validation:
  - The form validation has to keep values into fields so that in case of error the user doesn't need to rewrite everything into each field.
- Errors:
  - Errors have to warn the user properly with a red colored message of this format: "source [statusCode] : message". Source, statusCode and message are fields extracted from the payload returned by the API server. (Ref: above)

## [3.2]Search Query

- Method: GET
- URL: http://<domain>:8081/query
- parameters:

```
    limit: Joi.number().optional(),
     wordId: Joi.number().optional(),
     lema: Joi.string().optional(),
     pos: Joi.string().optional(),
     gerund: Joi.string().optional(),
     participle: Joi.string().optional(),
```

- Page specifications:
  - Choose whatever Html Object (INPUT seems good) to build the forms
  - “Search” button have to appear both at the top and bottom of all forms
- Form validation:
  - As described in current section “parameters” above
  - Of course every only one field at least need to be filled in order to launch the request
  - Null fields cannot appear in the request!
- Errors
  - Errors have to warn the user properly with a red colored message : “source [statusCode] : message” filled with respective payload parameters returned by the API server
  - If no results, the user has to be warned with a yellow colored message.
  - Error code list : above

## [3.3] View

This page is more an Angular.Js directive or something like that. It's supposed to draw the results from the API Server.

- Used context:
  - Search Query has to draw results this way
  - Add Query has to draw the results at the bottom of all forms so that user can check inserted values
- Data parsing:
  - The response from any Search Query is very verbose, you will draw only the fields appearing in the section “parameters” at Add Query (both optional and required)
  - The response from Add Query is very less verbose and can be parsed as stated above
- Page specifications:
  - There's no preferred Html Object to draw the fields, choose the best suited to support as readability as click-editable constraint described below.
- Actions:
  - Each drawn field will have to be click-editable
  - Each word will have his button “Update” on which an http update query will be bind
  - Fields in the list below must have a bootstrap “*glyphicon glyphicon-remove*” in front of its line so that user can delete it:
    - Synonym in Synonyms Array
    - Antonym Antonyms Array
    - Country in Countries Array
    - Definition in Definitions Array
      - Example in Example Array
    - Relative in Relatives Array
    - Hyperlink in Hyperlinks Array
  - On click on “Update” button, it must warn and ask about eventual deletions.

*note: Don't be surprised if there's not “Delete” query, update does the job.*

## [3.4] Update

- Method: POST
- URL: http://<domain>:8081/update
- parameters:

```
'wordId': Joi.number().required(),
  'gerund': Joi.string().optional(),
  'participle': Joi.string().optional(),
  'countries': Joi.array().includes(
    Joi.object().keys({
      'country' : Joi.string().min(3).max(32).required(),
      'frequency': Joi.number().min(0).max(100).integer().optional()
    }).required()
  ).optional(),
  'register': Joi.boolean().optional(),
  'language': Joi.string().optional(),
  'definitions': Joi.array().includes(
    Joi.object().keys({
        'id': Joi.number().required(),
        'definition': Joi.string().required(),
        'examples': Joi.array().includes(
          Joi.object().keys({
            'id': Joi.number().required(),
            'example' : Joi.string().required()
          }).required()
        ).optional()
    }).required()
  ).optional(),
  'synonyms': Joi.array().includes(
    Joi.object().keys({
        'id': Joi.number().required(),
        'lema': Joi.string().required(),
        'pos': Joi.string().required()
      }).required()
  ).optional(),
  'antonyms': Joi.array().includes(
    Joi.object().keys({
        'id': Joi.number().required(),
        'lema': Joi.string().required(),
        'pos': Joi.string().required()
      }).required()
  ).optional(),
  'relatives': Joi.array().includes(
    Joi.object().keys(
      {
        'id': Joi.number().required(),
        'lema': Joi.string().required(),
        'pos': Joi.string().required()
      }
    ).required()
  ).optional(),
  'hyperlinks': Joi.array().includes(
    Joi.object().keys({
      id: Joi.number().required(),
      hyperlink : Joi.string()..required()
    }).optional()
  )
```

*note: each field is optional (except "wordId"), you can have a basic update payload like that :*

```
{
        "wordId: 1,
        "gerund": "new value"
}
```

- Page specifications:
  - As described in the section above
- Errors
  - Errors have to warn the user properly with a red colored message : "source [statusCode] : message" filled with respective payload parameters returned by the API server
  - If no results, the user has to be warned with a yellow colored message.
  - Error code list : above

## [3.5] Basic Login Form

I would like to have a basic login form (only login, no register), a modal frame popping on requests would be great. I can add modification to the back-end if you need ("/login" page).

- Page specifications:
  - Login form
  - Password form
  - Validate Button

Tell me if you need anything

mail: amaury.brisou@gmail.com