1. API CALLS

1.1 Anatomy of a Grape API call

Grape may receive different type of API calls:

1. Filesystem request: the browser requests a file from the filesystem (for example an HTML, CSS, JS or image file)
2. Database API request: the API call is implemented as a database function, accepting and returning a JSON object
3. File download request: API calls providing a different result than JSON (for example access-controlled files). This calls starts with '/download'

1.1.1 FS Request

The first and most simple is a request for a file on the filesystem. A request that does not accept JSON, and does not start with /download, will fall under this category. This includes the initial call for index.html.

1.1.1.1 download_public_js_files

The **download_public_js_files** API call is a special API call that will traverse all subdirectories in the public directories (defined by ***public_directories***), with the names defined by ***compile_js_dirs***. The default values for ***compile_js_dirs*** is **pages**. This means all subdirectories named "pages" will be traversed for JS files, and served through this call.

1.1.2 DB API requests

Database API calls are the most commonly used API calls. The logic for the function is typically implemented as a function in PostgreSQL. The function being called in the database accepts a JSON parameter, and returns a JSON object with the result.

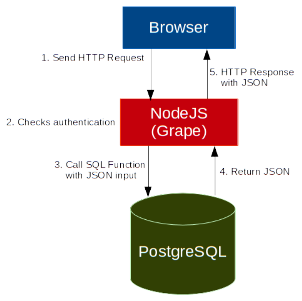

Fig. 1: Anatomy of a DB API

In order to create a DB API call, two changes are needed:

1. The API call needs to be registered in a JS file in one of the project's API directories (defined by the config option *api_directory*)

```
exports = module.exports = function(app) {
    // register the route
    app.get("/maths/sqrt/:value", api_maths_sqrt);
}
function api_maths_sqrt (req, res)
{
    // call the stored procedure for this API call
    res.locals.db.json_call("maths_sqrt", // the name of the PL/pgSQL function
        {value: req.params.value},    // Build the JSON object as input for this function
        null,                         // Optional callback (not used here)
        {response: res}               // Send the response to res
    );
}
```

2. A database function accepting a JSON input parameter and returning a JSON type must be defined in the database. Ideally, the API access function (accepting and returning a JSON) does not implement the business logic, but calls another SQL function to do this.

```
CREATE OR REPLACE FUNCTION maths_sqrt (JSON) RETURNS JSON AS $$
DECLARE
    _value NUMERIC;
    _result NUMERIC;
BEGIN

    _value := ($1->>'value')::NUMERIC; -- Extract values from JSON

    _result := sqrt(_value);           -- Calculation

    RETURN grape.api_success('result', _result); -- Build and return JSON object
END; $$ LANGUAGE plpgsql;
```

Note! API calls should always be properly documented! See the section "Documenting code" for more information on this

1.1.3 Other API calls

1.2 Access control

Access control is applied to all API calls. Before the API call is executed, the session is validated. If it cannot be validated, the default role **guest** is used. All users belongs to one or more roles, and always to the role named **all**. API calls are registered in the database (table access_path) by **path**, **method** and the **role** allowed. The **path** is a regular expression, matching the incoming URL of the request.

Note! The SQL function `grape.add_access_path (_path TEXT, _roles TEXT[], _methods TEXT[])` can be used to add a new access path.

1.3 Consuming an API call

In order to use an API call, you will need to know the following:

1. The URL of the call. This will look like a typical path, for example "/login"
2. The call method. This will usually be **GET** or **POST**