# Back end: main.py Documentation

`main.`**`add_module`**()

Add module to database. required in POST request body: SmObjId, PiqYear, PiqSession, whitelisted, searchterm

`main.`**`add_searchterm`**()

Add searchterm to DB, term is supplied in form data

`main.`**`admin`**()

Administrator front end view

`main.`**`app`** = *<Flask 'main'>*

`main.`**`check_which_saved`**(*modules: list*)

Check which modules are saved, and mark them as either white- or blacklisted accordingly

`main.`**`cors`** = *<flask_cors.extension.CORS object>*

`main.`**`db_config`** = *{'database': 'greenvvzdb', 'host': '127.0.0.1', 'password': 'greenvvzpw', 'user': 'greenvvz'}*

`main.`**`find_modules_for_course`**(*course: dict*)

Request detail page for course object, add Module subobjects(dicts) as list to given course object

`main.`**`find_studyprograms_for_module`**(*SmObjId: int*, *PiqYear: int*, *PiqSession: int*) → list

Request detail page for module object, add Studyprogrm subobjects(dicts) as list to given module obj

`main.`**`flag_module`**(*module_id: int*)

Flag saved module as whitelisted or blacklisted, depending on request.args.get('whitelisted')

`main.`**`get_blacklist`**()

`main.`**`get_modules`**(*whitelisted: bool*)

Get modules saved in the database, either blacklisted or whitelisted, as JSON response

`main.`**`get_searchterms`**()

get all search terms from DB

`main.`**`get_studyprograms`**()

Get distinct studyprograms associated with modules in the whitelist

`main.`**`get_studyprograms_modules`**()

Get Module-Studyprogramids assocations as a dictionary

`main.`**`get_whitelist`**()

`main.`**`hello_world`**()

Hello World test view

`main.`**`info`**()

Information about the API

`main.`**`public`**()

Public front end view

`main.`**`remove_module`**(*module_id: int*)

remove module from database by id

`main.`**`remove_searchterm`**(*searchterm_id: int*)
: remove searchterm from DB via id

`main.`**`require_appkey`**(*view_function*)
: decorator for checking the api-key, making unauthorized requests impossible

`main.`**`save_studyprograms_for_module`**(*module_id: int*, *studyprograms: list*)
: Save studyprogams for module in database, establish relationship

`main.`**`search`**()
: get modules based on search terms, marking those already on white- and blacklist

`main.`**`search_upwards`**()
: Find course matches, then find containing modules, # and containing study programs

`main.`**`update`**()
: Update saved modules to match their course catalogue counterparts, be there any changes

`main.`**`wrap_execute_for_modules_in_course`**(*course*)
: Wrapper function to be able to parallelize finding studyprograms for modules