Front end API

This page gathers all the API calls that can be used by the backend.

> Backend -> Front end

Warning

The functions can only be called if they are available on the `web/build` directory, therefore if you make a change using `npm run serve` won't show it, you will need to rebuild the front end with `npm run buildWeb` or by using `npm run start`.

Note

These functions can only be called after eel is initiated with `eel.init()`.

Core API

Collection of all functions/API calls available to the backend. You can find them in the `bolinho_api/core.py` file.

The JavaScript file can be found in the api folder.

ping()

ping()

Tries to ping the bolinho front-end, returns 1 if it worked

Python usage example

```
from bolinho_api.core import core_api

while True:
    try:
        if core_api.ping():
            print("got a ping!")
            break
        pass
    except:
        eel.sleep(1)
```

get_config_params()

get_config_params()

Tries to ping the bolinho front-end, returns 1 if it worked

Python usage example

```
from bolinho_api.core import core_api

config = core_api.get_config_params()
current_save_version = config["configVersion"]
print(current_save_version)
```

This function is located at `src/web/src/App.js`

go_to_experiment_page()

go_to_experiment_page()

Asks the front end to go to the experiment page.

Returns 1 if succeeded.

Python usage example

```
from bolinho_api.core import core_api

change_pages = True
if change_pages:
    core_api.go_to_experiment_page()
```

go_to_home_page()

go_to_home_page()

Asks the front end to go to the home page.

Returns 1 if succeeded.

Python usage example

```
from bolinho_api.core import core_api

change_pages = True
if change_pages:
    core_api.go_to_home_page()
```

set_is_connected()

set_is_connected()

Sets the variable "isConnected" on the front-end.

Python usage example

```
from bolinho_api.core import core_api

core_api.set_is_connected(True)
```

refresh_data()

refresh_data()

Sets the variable "isConnected" on the front-end.

Python usage example

```
from bolinho_api.core import core_api

add_material_to_db() #Arbitrary function that adds a material to the DB

core_api.refresh_data()
```

UI API

Collection of all functions/API calls available to the backend for UI in general. You can find them in the `bolinho_api/ui.py` file.

The JavaScript file can be found in the api folder.

success_alert(text)

success_alert(text)

Uses React-Toastify to create an success alert.

Python usage example

```
from bolinho_api.ui import ui_api

ui_api.success_alert("Success!")
```

error_alert(text)

error_alert(text)

Uses React-Toastify to create an error alert.

Python usage example

```
from bolinho_api.ui import ui_api

ui_api.error_alert("Error!")
```

loading_alert(text, callback_func)

loading_alert(text, callback_func)

Uses React-Toastify to create an loading alert.

Returns the Toast ID **Asynchronously** to a `callback_func`.

Must be used together with `update_alert`.

Python usage example

```
def save_and_end(toast_id):
    bolinho_app.end_experiment()
    run(bolinho_app.end_experiment())
    core_api.go_to_home_page()
    ui_api.update_alert("Salvo com sucesso!", True, toast_id)

ui_api.loading_alert("AGUARDE! Salvando no banco...", save_and_end)
```

update_alert(text, success, id)

update_alert(text, success, id)

Uses React-Toastify to update an existing alert.

If success is set to true it displays a success other wise shows an error

Python usage example

```
def save_and_end(toast_id):
    bolinho_app.end_experiment()
    run(bolinho_app.end_experiment())
    core_api.go_to_home_page()
    ui_api.update_alert("Salvo com sucesso!", True, toast_id)

ui_api.loading_alert("AGUARDE! Salvando no banco...", save_and_end)
```

prompt_user(description, options, callback_func)

prompt_user(description, options, callback_func)

Prompts the user with a 'description', and shows the 'options' to the user.

The result is passed to the callback_function

Python usage example

```
from bolinho_api.ui import ui_api

def get_result(result):
    if result == "yes":
        print("The user chose yes")
    print("The user chose no")

ui_api.prompt_user(
    description="Do you want to pay 1000?",
    options=["yes", "no"],
    callback_func= get_result,
    )
```

set_focus(focus_element: str)

error_alert(focus_element: str)

Focus in an specific element on the frontend.

WARNING Pay attention to the name of the element you are trying to focus

You can find them at https://github.com/HefestusTec/bolinho/blob/main/src/web/src/api/apiTypes.ts

Python usage example

```
from bolinho_api.ui import ui_api

ui_api.set_focus("connection-component")
```

Experiment page API

Collection of all functions/API calls available to the backend for the **experiment** routine. You can find them in the `bolinho_api/experiment.py` file.

The JavaScript file can be found at `web/src/api/contexts/ExperimentPageContext.tsx`.

set_time(newValue)

set_time(newValue)

Sets the current time of the experiment.

This variable is shown to the user as value and progress bar.

Python usage example

```
from bolinho_api.experiment import experiment_api

experiment_api.set_time(22)
```

set_delta_load(newValue)

set_delta_load(newValue)

Sets the current delta load.

This variable is shown to the user as value and progress bar.

Python usage example

```
from bolinho_api.experiment import experiment_api

experiment_api.set_delta_load(22)
```

get_readings()

get_readings()

Asks the front for the current Readings.

Returns an object of type Readings, this object gathers all the current readings of the machine. Such as Current z axis position, current load, and status

Python usage example

```
from bolinho_api.experiment import experiment_api

reading_obj = experiment_api.get_readings()

print(reading_obj.status)
```

set_readings(newValue)

set_readings(newValue)

Sets the current Readings.

Receives an object of type Readings, this object gathers all the current readings of the machine. Such as Current z axis position, current load, and status.

This function dumps the object to a JSON and sends it to the front end

Python usage example

```
from bolinho_api.experiment import experiment_api
from bolinho_api.classes import Readings

new_machine_readings = Readings(299, 87, 300, "not good")

experiment_api.set_readings(new_machine_readings)
```