Databanken en Webtechnologie: Project 21-22: API Documentation

Seppe Goossens

November 17, 2021

Contents

1 Show Classes

All the Show Classes mostly have the same general layout. They take some arguments and return a json with the corresponding values from the database. If the corresponding values are not in the database, the class returns a False statement. General show class layout:

```
class ShowData(Resource):
    def get(self, var1, var2):
        result_list = []
        q = "SELECT Col1, Col2 FROM Table WHERE Col1 = ? AND Col2 = ?"
        data = conn.execute(q,(var1, var2)).fetchall()
        for row in data:
            for item in row:
                result_list.append(item)
        if not result_list:
            return False
        else:
            return {"Col1": result_list[0], "Col2": result_list[1]}
```

1.1 Show log in data

Returns the corresponding username and password from the database. If the username and password don't correspond to the filled in username and password in the form, it returns a False statement.

- URL: http://127.0.0.1:8080/show/logindata/<string:username>,<string:password>
- Arguments:
  - <string:username>: email of the user, **type**: string
  - <string:password>: password of the user, **type**: string
- Return value:
  - if username and password are in the database: corresponding username and password, **type**: json
  - if username and password are not in the database: False statement, **type**: bool
- Method: GET

Call the class:

```
def check_login(username, password):
    url = api_address + "/show/logindata/" + username + "," + password
    data = requests.get(url)
    return data.json()
```

Output example:
{ "Username": "admin@vub.be", "Password": "admin" }

1.2 Show name and roll

Returns the corresponding name and roll of the user that is logging in.

- URL: http://127.0.0.1:8080/show/nameandroll/<string:username>
- Arguments:
  - <string:username>: email of the user, **type**: string
- Return value:
  - corresponding name and roll, **type**: json
- Method: GET

Call the class:

```python
def get_name_and_roll(username):
    url = api_address + "/show/nameandroll/" + username
    data = requests.get(url)
    return data.json()
```

Output example:
{ "Name": "admin", "Roll": "Admin" }

1.3 Show active user data

Returns the accountID, Email, Name and Roll of the current user.

- URL: http://127.0.0.1:8080/show/activeuserdata/<string:token>
- Arguments:
  - <string:token>: session key that is unique to the current user, **type**: string
- Return value:
  - json with the corresponding accountID, email, name and roll, **type**: json
- Method: GET

Call the class:

```python
def get_active_user_data(token):
    url = api_address + "/show/activeuserdata/" + token
    data = requests.get(url)
    return data.json()
```

Output example:
{"AccountID": 0, "Email": "admin@vub.be", "Name": "admin", "Roll": "Admin"}

1.4 Show presence per class

Returns the class name, name and email of the students that were present during a given class.

- URL: http://127.0.0.1:8080/show/presence/class/<string:class-name>
- Arguments:
  - <string:class-name>: name of the class, **type**: string
- Return value:
  - Json with the corresponding class name, name and email of the present students, **type**: json
- Method: GET

Call the class:

```
def get_present_students(class_name):
    url = api_address + "/show/presence/class/" + class_name
    data = requests.get(url)
    return data.json()
```

Output example:
{'Class Name': ['Basiswiskunde Les 1'], 'Name': ['Seppe Goossens'], 'Email': ['seppe.goossens@vub.be']}

1.5 Show duplicate presence codes

Returns a false statement if the given presence code is not in the database. Returns the presence code in the database if the given presence code exists in the database.

- URL: http://127.0.0.1:8080/show/duplicatecodes/<string:unique-code>
- Arguments:
  - <string:unique-code>: presence code of a given class, **type**: string
- Return value:
  - if presence code is in the database: Json with the corresponding presence code from the database, **type**: json
  - if presence code is not in the database: False statement, **type**: bool
- Method: GET

Call the class:

```
def check_duplicate_code(unique_code):
    url = api_address + "/show/duplicatecodes/" + unique_code
    data = requests.get(url)
    return data.json()
```

Output example:
{'Presence Code': 'AAAAA'}

1.6 Show course name

Returns the course name of a given course in the database. Returns a False statement if the course is not found in the database.

- URL: http://127.0.0.1:8080/show/course/<string:course-name>
- Arguments:
  - <string:course-name>: name of the course, **type**: string
- Return value:
  - if the name of the course is in the database: Json with the corresponding course name from the database, **type**: json
  - if name of the course is not in the database: False statement, **type**: bool
- Method: GET

Call the class:

```
def check_course_indb(course_name):
    url = api_address + "/show/course/" + course_name
    data = requests.get(url)
    return data.json()
```

Output example:
{'Course Name': 'Basiswiskunde'}

1.7 Show courseID for given course name

Returns courseID from the database for a given course name. Returns False statement if the courseID is found in the database

- URL: http://127.0.0.1:8080/show/courseID/<string:course-name>
- Arguments:
  - <string:course-name>: name of the course, **type**: string
- Return value:
  - if the name of the course is in the database: Json with the corresponding courseID from the database, **type**: json
  - if the name of the course is not in the database: False statement **type** bool
- Method: GET

Call the class:

```
def get_courseID(course_name):
    url= api_address + "/show/courseID/" + course_name
    data = requests.get(url)
    return data.json()
```

Output example:
{'courseID': '1'}

1.8 Show courseID for given class name

Returns courseID from the database for a given class name.

- URL: http://127.0.0.1:8080/show/courseID/class/<string:class-name>
- Arguments:
  - <string:class-name>: name of the class, **type**: string
- Return value:
  - corresponding courseID for the given class name **type**: json
- Method: GET

Call the class:

```python
def get_courseID_via_class_name(class_name):
    url= api_address + "/show/courseID/class/" + class_name
    data = requests.get(url)
    return data.json()
```

Output example:
{'courseID': '1'}

1.9 Show courseID for given presence code

Returns courseID from the database for a given presence code.

- URL: http://127.0.0.1:8080/show/courseID/presence/<string:presence-code>
- Arguments:
  - <string:presence-code>: unqiue presence code, **type**: string
- Return value:
  - corresponding courseID for the given presence code **type**: json
- Method: GET

Call the class:

```python
def get_courseID_via_presence_code(presence_code):
    url= api_address + "/show/courseID/presence/" + presence_code
    data = requests.get(url)
    return data.json()
```

Output example:
{'courseID': '1'}

1.10 Show class name

Returns the class name of a given class in the database. Returns a False statement if the class is not found in the database.

- URL: http://127.0.0.1:8080/show/class/<string:class-name>
- Arguments:
  - <string:class-name>: name of the class, **type**: string
- Return value:
  - if the name of the class is in the database: Json with the corresponding class name from the database, **type**: json
  - if name of the class is not in the database: False statement, **type**: bool
- Method: GET

Call the class:

```python
def check_class_indb(class_name):
    url = api_address + "/show/class/" + class_name
    data = requests.get(url)
    return data.json()
```

Output example:
{'Class Name': 'Basiswiskunde Les 1'}

1.11 Show username

Returns the username of a given username in the database. Returns a False statement if the username is not found in the database.

- URL: http://127.0.0.1:8080/show/user/<string:username>
- Arguments:
  - <string:username>: email of the user, **type**: string
- Return value:
  - if the username is in the database: Json with the corresponding username from the database, **type**: json
  - if username is not in the database: False statement, **type**: bool
- Method: GET

Call the class:

```python
def check_username_indb(username):
    url = api_address + "/show/user/" + username
    data = requests.get(url)
    return data.json()
```

Output example:
{'Email': 'admin@vub.be}

1.12 Show name, username and roll for given accountID

Returns the name, username and roll from the database for a given accountID.

- URL: http://127.0.0.1:8080/show/nameandusernameandroll/<int:accountID>
- Arguments:
  - <string:accountID>: unique accountID of the user, **type**: int
- Return value:
  - Json with the corresponding name, username and roll from the database, **type**: json
- Method: GET

Call the class:

```
def get_name_username_roll(accountID):
    url= api_address + "/show/nameandusernameandroll/" + accountID
    data = requests.get(url)
    return data.json()
```

Output example:
{'Name': 'admin', 'Username': 'admin@vub.be', 'Roll': 'admin'}

1.13 Show accountID for given username

Returns the accountID from the database for a given username.

- URL: http://127.0.0.1:8080/show/accountID/<string:username»
- Arguments:
  - <string:username>: email of the user, **type**: string
- Return value:
  - Json with the corresponding accountID from the database, **type**: json
- Method: GET

Call the class:

```
def get_accountID(username):
    url= api_address + "/show/accountID/" + username
    data = requests.get(url)
    return data.json()
```

Output example:
{'accountID': '1'}

1.14 Show all accountID's for a given courseID

Returns all the accountID's that are registrated for the given courseID.

- URL: http://127.0.0.1:8080/show/accountID/courseID/<int:courseID>
- Arguments:
  - <string:courseID>: courseID of the given course, **type**: int
- Return value:
  - Json with the corresponding accountID's from the database, **type**: json
- Method: GET

Call the class:

```
def get_accountID_via_courseID(courseID):
    url= api_address + "/show/accountID/" + courseID
    data = requests.get(url)
    return data.json()
```

Output example:
{'Account ID': [2, 1, 3]}

2 Add Classes

Like with the show classes, the add classes follow the same principle. They take some arguments and insert them into the database. All the add classes follow the same layout:

```
class AddToDatabase(Resource):
    def get(self, var1, var2):
        q = """INSERT OR IGNORE INTO "main"."Table" ("Col1", "Col2") VALUES (?, ?);"""
        conn.execute(q, (var1, var2))
        conn.commit()
```

2.1 Add active user data to database

Adds the username, name, roll and token of the active use into the active user table in the database.

- URL: http://127.0.0.1:8080/add/activeuser/
  <int:accountID>,<string:username>,<string:name>,<string:roll>,<string:token>
- Arguments:
  - <int:accountID>: accountID of the current user, **type**: int
  - <string:username>: email of the current user, **type**: string
  - <string:name>: name of the current user, **type**: string
  - <string:token>: unique session key, **type**: string

- Method: POST

Call the class:

```
def add_session_db(accountID, username, name, roll, token):
    url = '{}/add/activeuser/{},{},{},{},{}'
    .format(api_address, accountID, username, name, roll, token)
    requests.get(url)
```

2.2 Add active user to the presence table

Adds the accountID of the current user to the presence table.

- URL: http://127.0.0.1:8080/add/presence/<string:presence-code>,<int:account-ID>
- Arguments:
  - <string:presence-code>: unique presence code, **type**: string
  - <int:accountID>: accountID of the current user, **type**: int
- Method: POST

Call the class:

```
def add_presence_account_db(presence_code, account_ID):
    url = '{}/add/presence/{},{}'.format(api_address, presence_code, account_ID)
    requests.get(url)
```

2.3 Add course to the database

Adds the name of the course and the accountID that created it to the database.

- URL: http://127.0.0.1:8080/add/course/<string:course-name>,<int:accountID>
- Arguments:
  - <string:course-name>: name of the course, **type**: string
  - <int:accountID>: accountID of the current user, **type**: int
- Method: POST

Call the class:

```
def add_course_db(courseID, course_name, username):
    accountID_data = get_accountID(username)
    accountID = accountID_data['Account ID']
    url_add = '{}/add/assigncourse/{},{},{}'
    .format(api_address, courseID, course_name, accountID)
    requests.get(url_add)
```

2.4 Add class to the database

Adds the name of the class, courseID and the presence code to the database.

- URL: http://127.0.0.1:8080add/class/
  <string:class-name>,<int:courseID>,<string:presence-code>
- Arguments:
  - <string:class-name>: name of the class, **type**: string
  - <int:courseID>: corresponding courseID, **type**: int
  - <string:presence-code>: corresponding presence code, **type**: string
- Method: POST

Call the class:

```
def add_class_db(courseID, class_name, presence_code):
    url = '{}/add/assigncourse/{},{},{}'
    .format(api_address, courseID, class_name, presence_code)
    requests.get(url_add)
```

2.5 Add the assigned course to the database

Adds the course name and courseID to the database.

- URL: http://127.0.0.1:8080/add/assigncourse/<int:courseID>,<string:course-name>,<string:accountID>
- Arguments:
  - <int:courseID>: courseID of the course, **type**: int
  - <string:course-name>: name of the course, **type**: string
- Method: POST

Call the class:

```
def assign_course_db(courseID, course_name):
    url = '{}/add/assigncourse/{},{},{}'.format(api_address, courseID, course_name)
    requests.get(url)
```

3 Delete Class

This class deletes a certain record from a table. The layout is similar to the add class. Class layout:

```
class DeleteData(Resource):
    def get(self, var):
        q = "DELETE FROM Table WHERE Col = ?"
        conn.execute(q, (var, ))
        conn.commit()
```

3.1 Delete active user data

Deletes the accountID, email, name, roll name and session key of the active user from the active user table.

- URL: http://127.0.0.1:8080/delete/activeusers/<string:username>
- Arguments:
  - <string:username>: email of the user, **type**: string
- Method: POST

Call the class:

```
def delete_active_user(username):
    url = api_address + "/delete/activeusers/" + username
    requests.get(url)
```