Lab3: "REST API as integration"

Purpose

This labs purpose is to show how REST API's can be tested via webb requests, an important step in between unit testing individual classes and methds, and the step up to testing via any GUI.

The lab also introduces the concept of test fixtures in Pytest and how they can be used to structure the setup and teardown of the test environment.

Task description

The font-end communicates via a REST API with the database. In this lab the task will be to test the REST API sufficiently using Python, PyTest, and Requests.

Of course the API can be explored using extensions such as Postman, and should be provided as screenshots for extra points, but in the end a suite of automated Python tests should be implemented.

The implementation for the restapi.py can be found in the test system under:

`/home/pft/restapi/point-of-sale/restapi.py`

Write test cases that utilizes the REST API. The test cases to create should be:

1. Create a new Customer (without Equipment and SIM card).
2. Create a new SIM card.
3. Update Customer and the 'IMSIPtr' with the new SIM card ID.
4. Create a new Equipment, reference the already existing ProductID.
5. Update Customer and the IMEIPtr with the new EquipmentID.

The documentation for the REST API will need to be consulted.

Pytest - continued from Lab2

Builtin fixtures

Fixtures are a powerful way of replacing xUnit's seup and teardown methods. If your tests need to work on data you typically need to set them up. This is often a process that has to be repeated and independent for each test. This often leads to duplicate code.

The @pytest.fixture decorator provides an easy yet powerful way to setup and teardown resources. You can then pass these defined fixture objects into your test functions as input arguments.

```
#test_fixtures.py
import shutil

@pytest.fixture()
def db_setup():
    shutil.copy('clean_database.db', 'database.db')
```

Using a fixture in the test cases is simply a matter of passing it in as a function argument:

```
#test_cases.py
from test_fixtures import db_setup

def test_case_one(db_setup):
    ...

def test_case_two(db_setup):
    ...
```

Fixtures are implemented in a modular manner, as each fixture name triggers a fixture function which can itself use other fixtures. Ultimately we want each test to be independent, something that we can enforce by running your tests in random order. Fixtures helps making sure that a fixed baseline can be achieved.

How to write and run webb requests using Requests in Python

Making webb requests using the module "Requests" in Python (3rd party module separately installed) couldn't be easier.

```
>>> import requests
>>> response = requests.get('http://ifconfig.co/json')
>>> response.content
'{"ip":"196.196.30.68","ip_decimal":3301187140,"country":"United Kingdom","country_eu":true,
>>> type(response.content)
<type 'str'>
>>> type(response.json())
<type 'dict'>
>>> ifconf = response.json()
>>> ifconf['ip']
u'196.196.30.68'
>>> response.status_code
200
>>> response.ok
True
```

Requests also supports other methods to make webb requests:

```
>>> r = requests.get('https://api.github.com/events')
>>> r = requests.post("http://httpbin.org/post")
>>> r = requests.put("http://httpbin.org/put")
>>> r = requests.delete("http://httpbin.org/delete")
```

And including parameters and/or data into your requests is done in the following way:

```
>>> payload = {'key1': 'value1', 'key2': 'value2'}
>>> r = requests.get("http://httpbin.org/get", params=payload)
>>> print(r.url)
http://httpbin.org/get?key2=value2&key1=value1
>>>
>>> image_data = open('smelly_cat.jpeg', 'rb').read()
>>> r = requests.get("http://other.com/get", data=image_data)
>>> r = requests.get("http://other.com/get", data=json.dumps(payload))
>>> r = requests.get("http://other.com/get", json=payload)
```

Notice the convenience parameter "json" that can be used directly, instead of having to use the "data"-parameter and the need to convert the input to a valid json-format first.

The full documentation and description of "Requests" can be found at: http://docs.python-requests.org/en/master/