py-ispyb : current status and future perspectives

Ivars Karpičs

EMBL

Content

- py-ispyb
  - Overview
  - Structure: plug-in extension and modules, data models and schemas.
  - Site specific configuration: authentication and authorization
  - Available services and routes
  - Testing, continues integration and deployment
- Road map

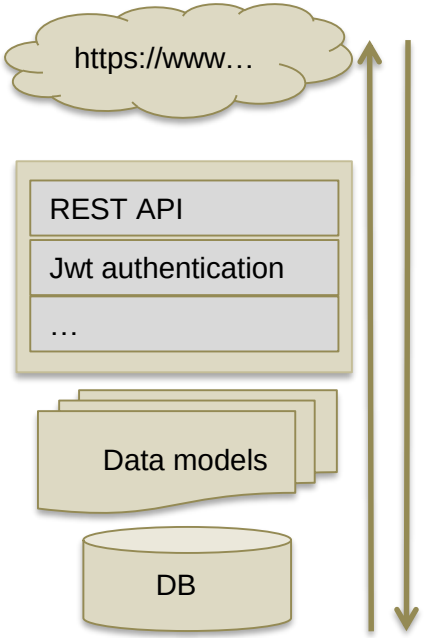
Overview

- **Python 3.x**
- **Flask**>=1.1,<2
- **flask-restx**
- Flask-Cors>=3.0.8,<4

- **SQLAlchemy**>=1.3.0,<2
- Flask-SQLAlchemy>=2.4,<3

- **marshmallow**>=2.13.5,<3
- flask-marshmallow>=0.7,<0.8
- marshmallow-sqlalchemy>=0.12,<0.13
- marshmallow_jsonschema

- **Pyjwt**
- gunicorn
- mysqlclient



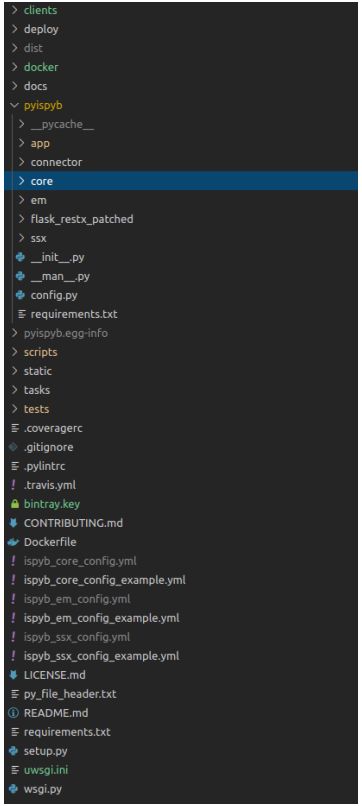
Structure

- System related **extensions**: api, auth, db, logging.
- Service definition directories:
  - core
  - ssx
  - em

Each service contains:

- **models.py**: sqlalchemy classes.
- **schemas**: flask, marshmallow and json schemas.
- **modules**: data handling modules.
- **routes**: end point definition

- Scripts, tests, deploy, docker, setup.py



Authentication and authorization

- Configuration defined in the **`config.py`**.
- Site specific configuration yml file can override parameters in the config.py.
- **`ispyb_core_config_example.yml`** as an example is available in git.
- Authentication module and class are defined in the config and loaded dynamically.
- **`AbstractAuth`** class defines interface and **`EMBLAuth`** as an example using LDAP is available.
- Authorization is also defined in the config and defines which user roles can access resources (endpoints).
- Decorators used in the route definition.

ispyb_core_config_example.yml

```yaml
server:
    SERVICE_NAME : "core"
    API_ROOT : "/ispyb/api/v1"
    SQLALCHEMY_DATABASE_URI : "mysql://mxuser:mxpass@localhost/pydb_test"

    AUTH_MODULE : "pyispyb.app.extensions.auth.DummyAuth"
    AUTH_CLASS : "DummyAuth"
    MASTER_TOKEN : "MasterToken"

authorization_rules:
    proposals : {
        "get": ["manager", "admin", "user"],
        "post": ["manager", "admin"]
    }
```

```python
@api.route("/<int:proposal_id>", endpoint="proposal_by_id")
@api.param("proposal_id", "Proposal id (integer)")
@api.doc(security="apikey")
@api.response(code=HTTPStatus.NOT_FOUND, description="Proposal not found.")
class ProposalById(Resource):

    """Allows to get/set/delete a proposal"""

    @api.doc(description="proposal_id should be an integer ")
    @api.marshal_with(proposal_schemas.f_schema, skip_none=False, code=HTTPStatus.OK)
    @token_required
    @authorization_required
    def get(self, proposal_id):
        """Returns a proposal by proposalId"""
        return proposal.get_proposal_by_id(proposal_id)

    @api.expect(proposal_schemas.f_schema)
    @api.marshal_with(proposal_schemas.f_schema, code=HTTPStatus.CREATED)
    @token_required
    @authorization_required
    def put(self, proposal_id):
        """Fully updates proposal with id proposal_id"""
        current_app.logger.info("Update proposal %d" % proposal_id)
        return proposal.update_proposal(proposal_id, api.payload)
```

Available routes

- Reduced amount of endpoints reduces the complexity of api.
- Instead of several endpoints use one endpoint with query arguments to retrieve data.

```
<xs:element name="findProposal" type="tns:findProposal"/>
<xs:element name="findProposalByCodeAndNumber" type="tns:findProposalByCodeAndNumber"/>
<xs:element name="findProposalByCodeAndNumberResponse" type="tns:findProposalByCodeAndNumberResponse"/>
<xs:element name="findProposalByLoginAndBeamline" type="tns:findProposalByLoginAndBeamline"/>
<xs:element name="findProposalByLoginAndBeamlineResponse" type="tns:findProposalByLoginAndBeamlineResponse"/>
<xs:element name="findProposalResponse" type="tns:findProposalResponse"/>
<xs:element name="findProposalsByLoginName" type="tns:findProposalsByLoginName"/>
<xs:element name="findProposalsByLoginNameResponse" type="tns:findProposalsByLoginNameResponse"/>
```

```
/proposals:                                {…}
/proposals/{proposal_id}:                  {…}
/proposals/{proposal_id}/info:             {…}
```

Method GET URL http://127.0.0.1:5000/ispyb/api/v1/proposals?proposalCode=cm&proposalNumber=14451

Headers

Authorization: Bearer MasterToken

Body

Request Body

[-] Response

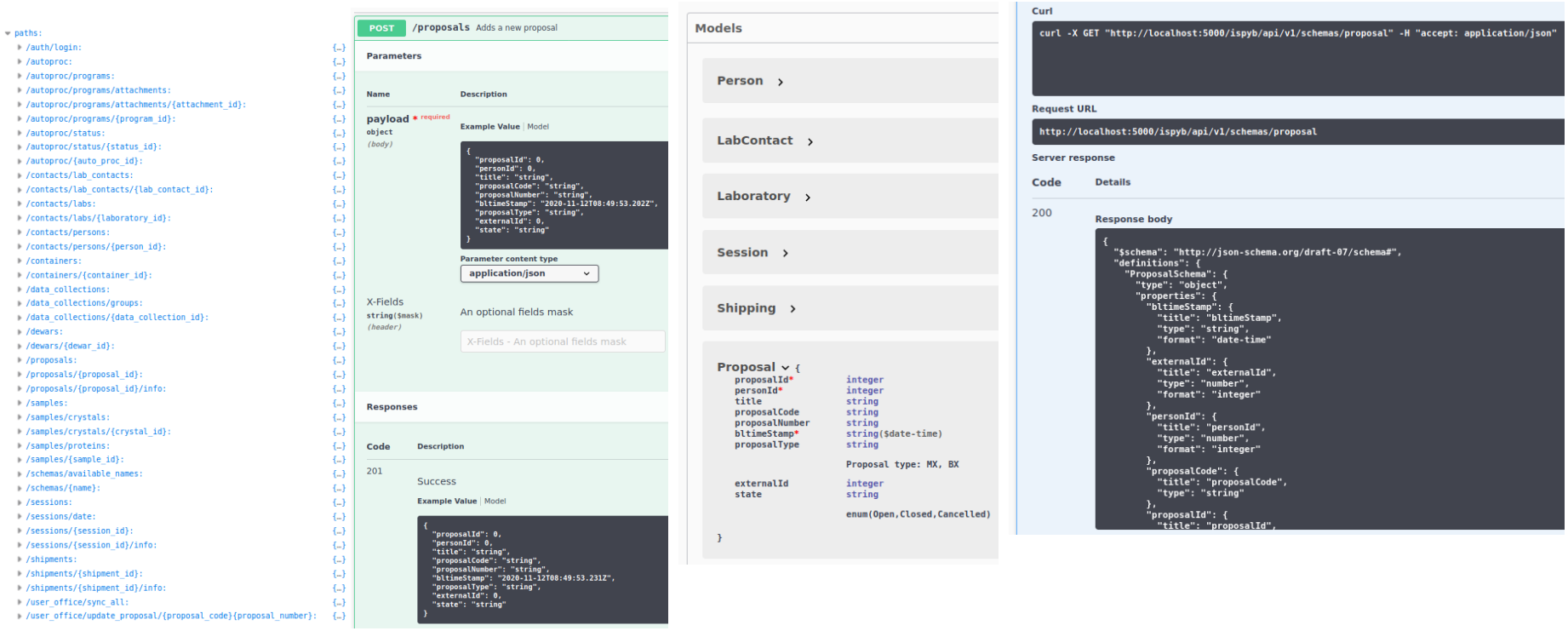
Headers Response Preview


```
{
  "data": {
    "total": 4,
    "rows": [{
      "proposalCode": "cm",
      "state": "Open",
      "externalId": null,
      "title": "I03 Commissioning Directory 2016",
      "proposalId": 37027,
      "personId": 1,
      "bltimeStamp": "2015-12-21T15:20:43+00:00",
      "proposalType": null,
      "proposalNumber": "14451"
    }]
  },
  "message": null
}
```


EMBL

Available routes

- Swagger ui to document and test api.



Continues integration and deployment

- Testing and continues integration with Travis ensure the api consistency.

```yaml
language: python

matrix:
  include:
    - python: 3.7
      #    - python: pypy
      #env: OPTIONAL=1

addons:
  mariadb: 10.3


install:
        - sudo apt-get install -y python3-mysqldb
        - pip install -r requirements.txt
        - pip install -r tests/requirements.txt

script:
        - cp ispyb_core_config_example.yml ispyb_core_config.yml
        - cp ispyb_ssx_config_example.yml ispyb_ssx_config.yml
        - cd scripts
        - bash create_core_db.sh
        - bash create_ssx_db.sh
        - bash run_all.sh
        - cd ..
        - pylint -E app ispyb_core ispyb_ssx
        - coverage run -m pytest tests/core
        - bash <(curl -s https://codecov.io/bash)
        - coverage run -m pytest tests/ssx
        - bash <(curl -s https://codecov.io/bash)
```

```
The command "pylint -E app ispyb_core ispyb_ssx" exited with 0.
$ coverage run -m pytest tests/core
============================= test session starts ==============================
platform linux -- Python 3.7.1, pytest-5.4.3, py-1.8.0, pluggy-0.12.0
rootdir: /home/travis/build/ispyb/py-ispyb
plugins: cov-2.5.1
collected 11 items

tests/core/functional/test_a_post.py .                                   [  9%]
tests/core/functional/test_b_get.py .                                    [ 18%]
tests/core/functional/test_c_patch.py .                                  [ 27%]
tests/core/functional/test_d_delete.py .                                 [ 36%]
tests/core/unit/test_models.py .......                                   [100%]

============================== warnings summary ================================
tests/core/functional/test_a_post.py::test_post
  /home/travis/virtualenv/python3.7.1/lib/python3.7/distutils/__init__.py:4: Depr
in favour of importlib; see the module's documentation for alternative uses
    import imp

-- Docs: https://docs.pytest.org/en/latest/warnings.html
======================= 11 passed, 1 warning in 2.44s ========================
The command "coverage run -m pytest tests/core" exited with 0.
```

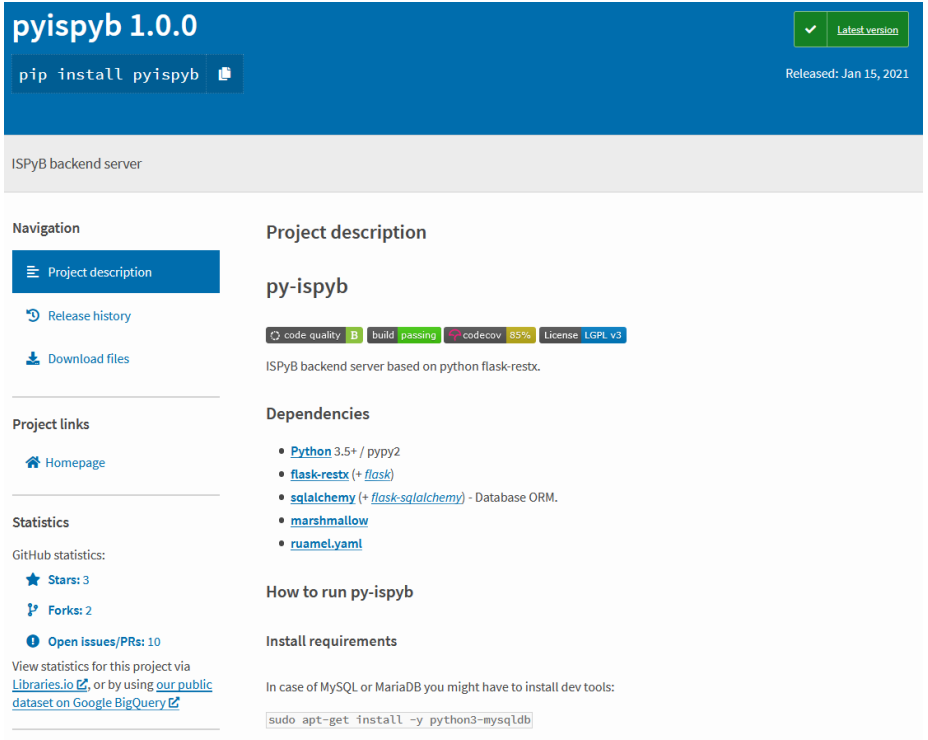
Versioning and deployment

```
pip install pyispyb

from pyispyb import create_app
ispyb_app = create_app(config_filename, 'dev')
ispyb_app.run(host='0.0.0.0', port=5000, debug=True)


gunicorn –b 127.0.0.0:4000 "app:create_app()"


from gevent.pywsgi import WSGIServer
from pyispyb import create_app
ispyb_app = create_app(config_filename, 'dev')
ispyb_server = WSGIServer(('', 5000), ispyb_app)
ispyb_server.serve_forever()
```



Road map

- Feb. 2020 – strategy meeting at EMBL Hamburg. Goal to develop a new backend for ispyb.
- Jun. 2020 – first version of py-ispyb presented. Contains one core service.
- Nov. 2020 – extended set of routes and SSX service. Microservice management with Kong.
- May 2021 – cryoem service. First release and package at pypi.

- Adapting existing guis EXI2 and synchweb to the py-ispyb backend.

How to contribute and collaborate

- Check the contributing guidelines.
- Developers code camp.
- Use more agile methods: sprints, pair programming and review.
- Lets collaborate…

Thank you for your attention!