
ESRF
The European Synchrotron

# py-ISPyB
# New version proposal

# Our goal with this version

## Validate usability in real world use cases

- *Only very simple/naive requests implemented in the previous version*

## Implement a minimal viable version for CryoEM

- *Minimal codebase*
  - *Removed all unnecessary code, can be reintegrated later on*
  - *Makes it easier to get started with*
- *Prepares the integration of Serial Crystallography*

## Improve/implement some necessary functionalities

- *Authentication*
- *Authorization*
- *Backward compatibility*

# Database access performance

# Database access performance

**Three ways to get DB data with SQLAlchemy:**

- *Single Entity*
- *Joined load*
- *SQL*

# Database access performance

## *Single Entity*

**Pros:**
- *clear and **simple** for simple requests*

**Cons:**
- *very **verbose** for complicated requests*
- *generates lots of **small requests** to DB*

## *Joined load*

**Pros:**
- *only deal with **model***
- *better code **maintainability***

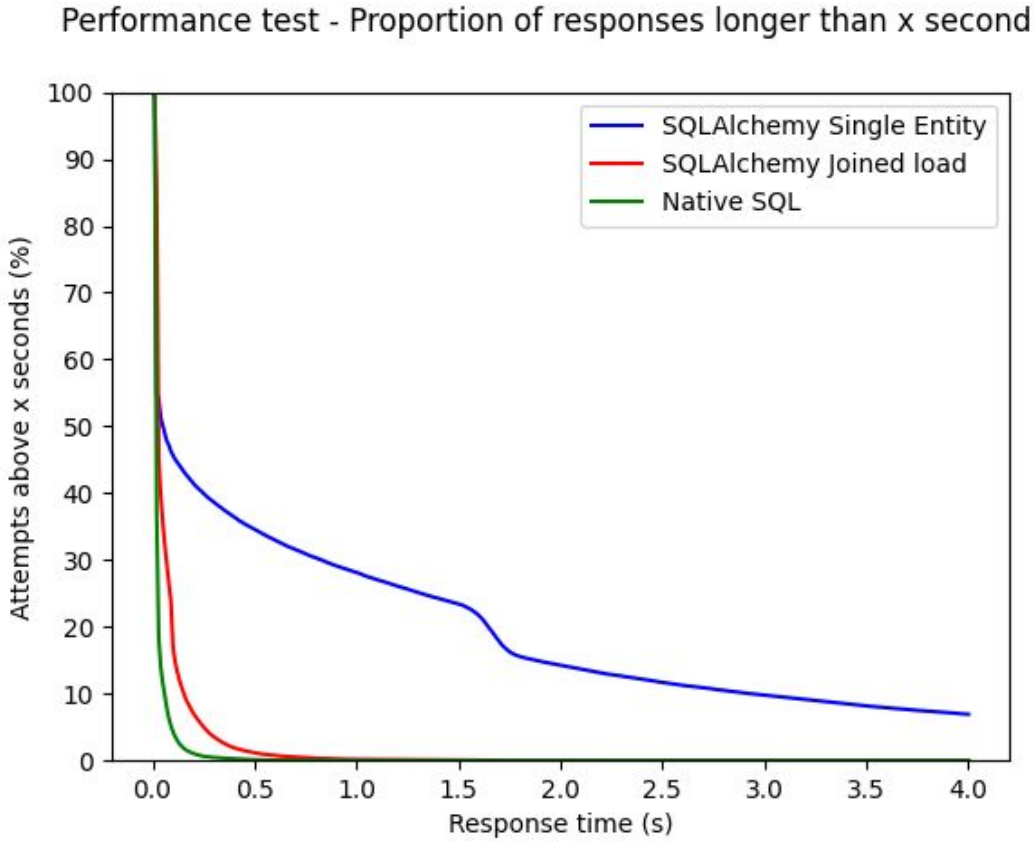
**Cons:**
- *known for slightly **lower performances***

## *SQL*

**Pros:**
- ***fast** execution if well written*
- *requests **already exist** in java*

**Cons:**
- *hard to read & **low maintainability***

# Database access performance

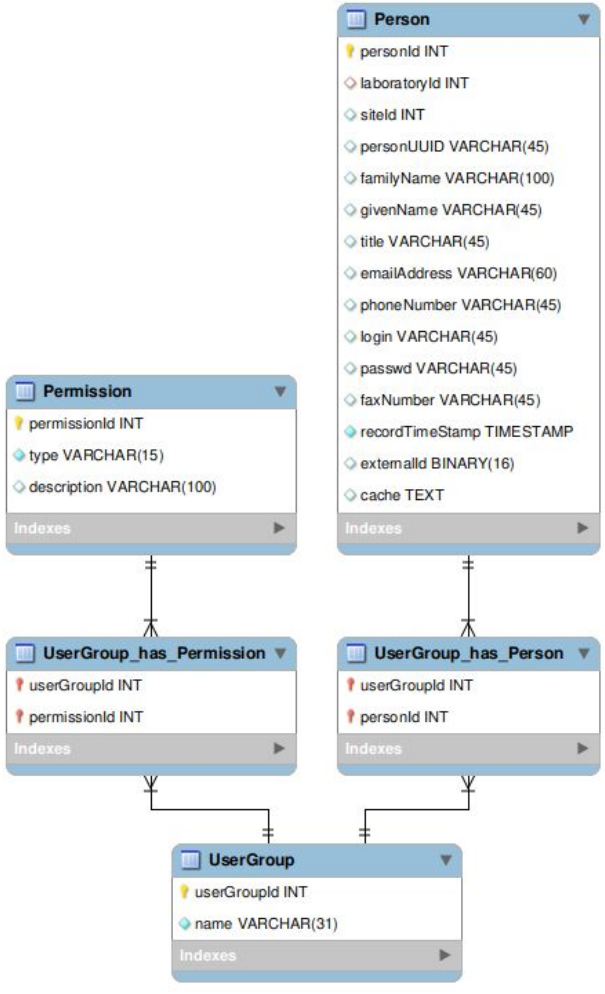



**Initial approach:**

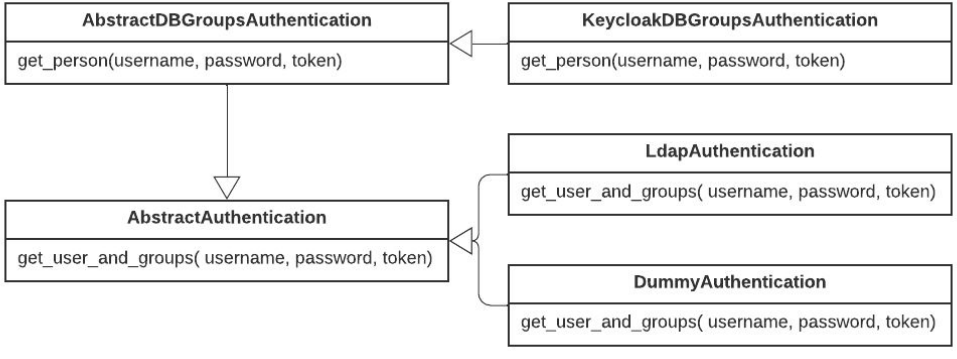
- *re-use already written **SQL** requests from ISPyB for existing endpoints (i.e. MX/EM)*
- *Use **SQLAlchemy joined load** for new developments (i.e. SSX)*

# Developments

# Authentication




- Using DB Group & Permission
- Multiple authentication mechanisms
  - Natively supports:
    - LDAP
    - Keycloak
    - Dummy (For developments)
  - Possibility to add your own auth ***via plugin***



# *Authentication* - **configuration**

```
AUTH:
  - keycloak:
      ENABLED: true
      AUTH_MODULE: "pyispyb.app.extensions.auth.KeycloakDBGroupsAuthentication"
      AUTH_CLASS: "KeycloakDBGroupsAuthentication"
      CONFIG:
        KEYCLOAK_SERVER_URL: "your_server"
        KEYCLOAK_CLIENT_ID: "your_client"
        KEYCLOAK_REALM_NAME: "your_realm"
        KEYCLOAK_CLIENT_SECRET_KEY: "your_secret"
  - ldap:
      ENABLED: true
      AUTH_MODULE: "pyispyb.app.extensions.auth.LdapAuthentication"
      AUTH_CLASS: "LdapAuthentication"
      CONFIG:
        LDAP_URI: "ldap://your_ldap"
        LDAP_BASE_INTERNAL: "ou=People,dc=esrf,dc=fr"
        LDAP_BASE_EXTERNAL: "ou=Pxwebgroups,dc=esrf,dc=fr"
  - dummy: # /!\/!\/!\ ONLY USE FOR TESTS /!\/!\/!\
      ENABLED: false
      AUTH_MODULE: "pyispyb.app.extensions.auth.DummyAuthentication"
      AUTH_CLASS: "DummyAuthentication"
```

## Authorization system

- `@authentication_required`
  - *only accessible to **authenticated users** (no permissions checking)*
- `@permission_required`
  - *Specify list of required permission(s)*
- `@proposal_authorization_required`
  - *Permission **all_proposals***
  - *Permission **own_proposals** and access to the specified proposal*
- `@session_authorization_required`
  - *Permission **all_sessions***
  - *Permission **own_sessions** and access to the specified session*

# Backward compatibility

## Legacy routes

```
New route      ➤     @api.route("/groups/session/<int:session_id>")
Legacy Route   ➤     @legacy_api.route("/<token>/proposal/session/<session_id>/list")
```

- Enables seamless compatibility with tools based on ISPyB (EXI, EXI2…) -> faster adoption
- Accessible with `/legacy` prefix
- Legacy endpoints can be accessed through both new and legacy routes
- Leaves room for more modern future developments on new endpoints

# Alternatives

# Technical improvements

- **Python typing**
- **Improved documentation**
  - *Endpoint parameter types*
  - *Value constraints*
- **Repeated model definition**
  - *flask-restx*
  - *marshmallow*

**Fast-api ?**