ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA115

FROM MXCuBE3 TO BSXCuBE3 A WEB APPLICATION FOR BioSAXS EXPERIMENT CONTROL

M. Oskarsson, A. Beteva, D. D. S. De Sanctis, S. Fisher, G. Leonard, P. Pernot, M. D. Tully
ESRF, Grenoble, France
J. B. Florial, A. A. McCarthy, EMBL Grenoble Outstation, France

Abstract

A new version of the beamline control application BSXCuBE (BioSAXS Customized Beamline Environment) designed to control BioSAXS experiments at the new ESRF Extremely Brilliant Source (EBS) is under development. The new application is implemented as a Web application and it is based on MXCuBE3 (Macromolecular Crystallography Customized Beamline Environment version 3) from which inherits the same technology stack and application structure. This approach allows for faster development and easier maintenance. The advances in architecture and the design of new features in BSXCuBE3 are intended to enhance the automation on BioSAXS beamlines and facilitate the integration of new sample setups, such as microfluidics. As for MXCuBE3, the access to the application from any web browser natively allows the execution of remote experiments. Moreover, the ergonomics of the interface further simplifies beamline operation even for non-experienced users. This work presents the current status of BSXCuBE3 and demonstrates how the development of MXCuBE3 has contributed to the construction of a BioSAXS application.

BACKGROUND

MXCuBE3 (Macromolecular Crystallography Customized Beamline Environment version 3) is the latest generation of a beamline control application allowing beamline users to carry out experiments in Macromolecular Crystallography (MX). Originally designed and developed at the ESRF [1] the MXCuBE project has evolved to become a collaborative development which now involves eleven institutes [2]

MXCuBE3 is currently in production at the ESRF MX-beamlines ID29 [3] and ID23-2 [4], BioMAX at MAX-IV [5] and XRD2 at Elettra [6]. The application has also been installed for testing on three other ESRF MX beamlines ID30A-1, ID30A-3 and ID30B, with the aim of full deployment after the ESRF-EBS upgrade [7]. MXCuBE3 is built on well-established libraries for web development, including React [8], Redux [9], SocketIO [10] and React-Bootstrap, in order to create an intuitive, user-friendly application [11].

First released in 2005 MXCuBE [12] is now the most frequently used software for MX experiment control and data acquisition in Europe. The user experience of the application has always been important and has gained more focus in recent years, making the application easier to use such that the user can focus on the experiment at hand rather than the complexities of the beamline hardware.

The success of MXCuBE3 has inspired and influenced the development of a new general framework for beamline control applications, capable of serving both web and Qt front ends. This framework consists of reusable UI components, many already existing in MXCuBE3, and a general purpose backend. The framework further facilitates good development practices by providing patterns and abstractions for both back-end and front-end development. This design effectively allows different applications to share the same application logic regardless of the frontend technology used.

One of the first applications to be implemented in this new framework is an experiment control application for BioSAXS experiments, BSXCuBE3, scheduled for deployment in August 2020. BioSAXS is often used as a complementary technique to MX and many potential BSXCuBE3 users are already familiar with the MXCuBE3 user interface (UI). Building the BSXCuBE3 UI with the same technology and design as MXCuBE3 thus not only shortens development time and eases maintenance, it also makes the two interfaces mutually consistent from a user point of view.

GENERAL APPLICATION FRAMEWORK

Many beamline control applications at ESRF have been successfully developed and deployed using the ESRF FWK2, a Qt3 based general purpose beamline application framework [13]. However, the EBS upgrade program, the introduction of the BLISS beamline control system [14] and the obsolescence of Qt3 required the introduction of a new application framework. MXCuBE3 was developed as a web application to facilitate remote access experiments at MX beamlines, while most other ESRF beamline control applications are designed with Qt. The structure of the new application framework makes it possible to use the same back-end for both Qt and Web front-end technologies. While MXCuBE3 was developed on top of the MXCuBE2 back-end which is today exclusively used for MX experiments, BSXCuBE3 shares the same back-end solution with a larger set of beamline applications, further decreasing the maintenance and development time.

Back-end

The back-end of the application framework (Fig. 1) is written in Python 3 and can be divided into four main parts: Web server, Control system integration, Common application logic, and Application specific components.

Web Server As is the case for MXCuBE3, the BSXCuBE3 web server is based on a Web Server Gateway

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA115

Interface (WSGI) compatible micro web framework called Flask [15]. A library called Socket.IO is used to provide bidirectional asynchronous communications such as events. The web server exposes a REST API for the clients (Web or Qt) to the underlying framework and beamline control system. Tools for validation and marshalling of the REST resources are built on top of Marshmallow [16] and apispec [17] which also provides Swagger [18] documentation.

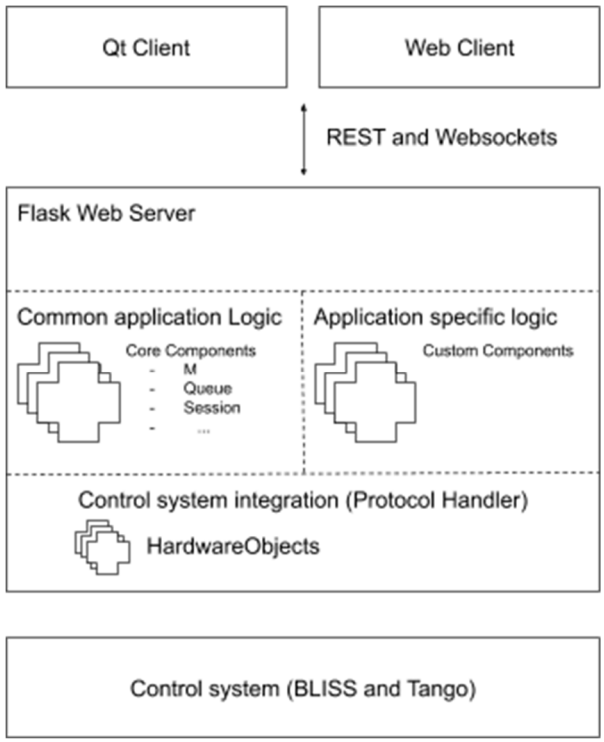

Figure 1: Backend overview.

Control system integration The framework developed provides the means to integrate with beamline control systems through a ProtocolHandler. Here, the underlying instrumentation is made available to the framework through HardwareObjects which, together with abstract interfaces and mappings, provide a coherent interface to the other components of the application. The framework currently implements ProtocolHandlers for BLISS, Tango, and FWK2 HardwareRepository.

Common application logic There are a few concepts that are common to many beamline applications including:

- Beamline calibration - for storing beamline calibration data
- Session - Handling data associated with the users currently logged in
- Authentication - For user authentication
- Microscope viewer - video stream with the possibility to mark regions and points of interest on the sample
- Task queue - for queuing tasks to be performed
- LIMS - integration with Laboratory Information Management System and meta data handling.
- Schema definition - For defining data models that can be used on both client and server for input validation and generic form display.

These concepts are implemented as Core Components and are in general configurable through YML files.

Application specific logic The framework can be extended with Components to implement application specific logic.

Front-end - Web Client

Similarly to MXCuBE3, the BSXCuBE3 UI is implemented with React, Redux and Bootstrap using HTML5 and ECMAScript 9. React provides the means to encapsulate interaction logic and to create reusable UI components. The components are expressed in Javascript XML (JSX) [19] a syntactical extension to Javascript which adds the possibility to use a HTML-like syntax together with Javascript. This makes it possible to easily share UI components between the various applications.

A standard build tool chain very similar to the one adopted in MXCuBE3 [20] is used, it is uses Webpack and babel to bundle the various assets and compile the sources down to ECMAScript 5, which has more consistent support across browsers. This allows the developers to use the latest features of the Javascript language and still keep cross browser compatibility.

Data provided to a UI component are handled with the state management library Redux. Redux keeps the entire application state in a 'store'. The Redux store is an immutable data structure which can only be updated by dispatching a Redux-action. A Redux-action is an object that contains the arguments for the state transfer. The state transformation itself is defined by a pure function called a reducer that takes the current state and the Redux-action as parameters and returns the new state (Fig. 2).

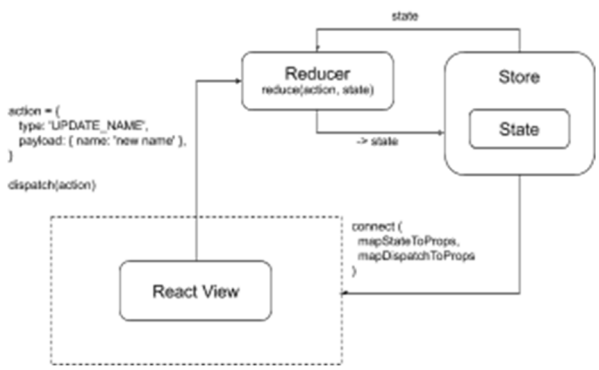

Figure 2: React-Redux update cycle.

Both React and Redux rely on shallow comparison through reference equality to handle updates. A challenge for developers can be to maintain the immutability contract that comes with Redux. Failing to do so will cause the shallow comparison to fail and the component displaying the data is not updated correctly as a result. There are a few libraries that aim to help the developer writing the reducing functions, among others: Immutable.JS [21] seamless-immutable [22] and ImmerJS [23].

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA115

The Redux reducer is often written as a function containing a switch statement. This switch statement can often become large and difficult to manage especially if native javascript operations like Object.assign and the spread operator is used for the state transformation.

The front-end part of the general application framework provides an abstraction for writing the Redux reducers, actions and selectors. There are corresponding base classes for each concept called Actions, Reducers and Selectors, these are combined into a single entity by a class named Provider.

The Reducer object makes it possible to write the reducer as an object where each action corresponds to a member function. The abstraction further provides helper functions, currently wrapping seamless-immutable, to handle the state transformation. The idea of the Provider concept is to reduce risk for mistakes with state transformation, encapsulating the reducer and corresponding actions so that they can be easily reused.

Communication with the back-end is done via Asynchronous JavaScript And XML (AJAX) requests and websockets. Helper classes wrapping the libraries Axios [24] and socket.io-client [25] are used to implement token based authentication.

User Interface

Both MXCuBE3 and BSXCuBE3 have the same key concepts such as beamline configuration and calibration, Microscope viewer, a queuing system, one or several types of sample changers and LIMS integration. This makes it possible for the two applications to have the same general look and feel. Indeed, many of the graphical components developed for MXCuBE3 can be directly reused in BSXCuBE3.

The overall layout of BSXCuBE3 (Figs. 3 and 4) is very similar to that of MXCuBE3 with navigation and general beamline information at the top of the application. The user is faced with a main menu with an item for each experiment mode. At the time of writing the different experiment modes foreseen are: Sample Changer, HPLC, Scan and Workflow.

- Sample Changer - Experiments using an Arinax BioSAXS sample changer [26]
- HPLC - Experiments using a High-performance liquid chromatography (HPLC) device
- Scan - Scans on points or regions of interest on, for instance, microfluidic chips
- Workflow - A way to create a custom collection sequence by queuing various predefined scripts in combination with the experiment types above.

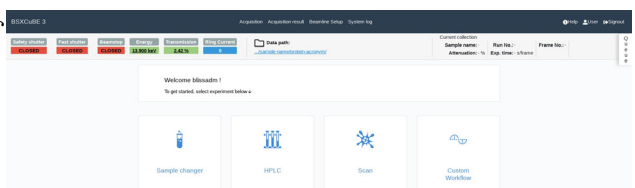

Figure 3: BSXCuBE Welcome page.

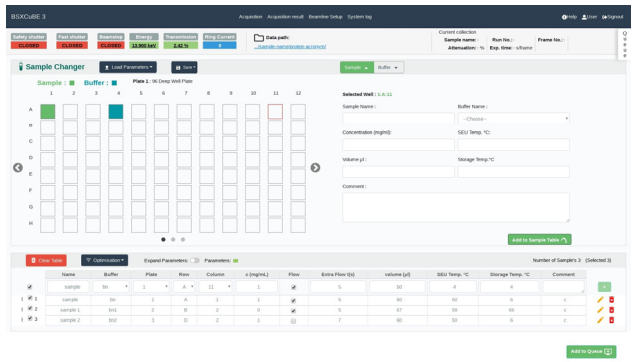

Figure 4: BSXCuBE Sample Changer experiment.

CONCLUSION AND FUTURE WORK

The technology stack and overall application structure of BSXCuBE3 is very similar to that of MXCuBE3, the development of which has inspired and influenced a new ESRF application framework. While MXCuBE3 uses tailored backend solution BSXCuBE3 benefits from the development of this new framework. Relying on a backend that will be common for all ESRF beamline control applications will decrease maintenance and development time as well as provide better operational support. Further, the user experience developed in MXCuBE3 can further be easily reused in BSXCuBE3, a particular asset as the two applications share a common user community.

ACRONYMS

BSXCuBE: BioSAXS Customized Beamline Environment
BioSAXS: Biological small angle X-ray scattering
MXCuBE3: Macromolecular Xtallography Customized Beamline Environment version 3
MX: Macromolecular Xtallography
FWK2: Framework 2
WSGI: Web Server Gateway Interface
REST: Representational State Transfer
API: Application Program Interface
OAV: On axis viewer
LIMS: Laboratory InforMation System
XML: eXtensible Markup Language
JSX: JavaScript XML
HTML: Hypertext Markup Language
AJAX: Asynchronous JavaScript And XML
HPLC: High-performance liquid chromatography

REFERENCES

[1] U. Mueller *et al.*, "MXCuBE3: A New Era of MX-Beamline Control Begins," *Synchrotron Radiat. News*, vol. 30, no. 1, pp. 22-27, Jan. 2017.

[2] M. Oscarsson *et al.*, "*MXCuBE2*: the dawn of *MXCuBE* Collaboration," *J. Synchrotron Radiat.*, vol. 26, no. 2, pp. 393-405, Mar. 2019.

[3] D. de Sanctis *et al.*, "ID29: a high-intensity highly automated ESRF beamline for macromolecular crystallography experiments exploiting anomalous scattering," *J. Synchrotron Radiat.*, vol. 19, no. pt 3, pp. 455-461, May 2012.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA115

[4] D. Flot *et al.*, "The ID23-2 structural biology microfocus beamline at the ESRF," *J. Synchrotron Radiat.*, vol. 17, no. 1, pp. 107-118, Jan. 2010.

[5] M. M. G. Thunnisen *et al.*, "BioMAX: The Future Macromolecular Crystallography Beamline at MAX IV," *J. Phys. Conf. Ser.*, vol. 425, no. 7, p. 072012, Mar. 2013.

[6] A. Lausi *et al.*, "Status of the crystallography beamlines at Elettra," *The European Physical Journal Plus*, vol. 130, no. 3, p. 43, Mar. 2015.

[7] ESRF-EBS, http://www.esrf.fr/about/upgrade

[8] reactjs, https://reactjs.org/.

[9] Redux, https://redux.js.org/.

[10] socket.io, https://socket.io/.

[11] React Bootstrap, https://react-bootstrap.github.io/.

[12] J. Gabadinho *et al.*, "*MxCuBE*: a synchrotron beamline control environment customized for macromolecular crystallography experiments," *J. Synchrotron Radiat.*, vol. 17, no. 5, pp. 700–707, Sep. 2010.

[13] M. Guijarro, G. Berruyer, J. Klora, and V. Rey-Bakaikoa, "The Bliss Framework project," in *Proc. Nobugs 2004 Conference,* Villigen PSI, Switzerland, paper 00065, 2004.

[14] Bliss, https://gitlab.esrf.fr/bliss

[15] Flask, https://palletsprojects.com/p/flask/.

[16] marshmallow, https://marshmallow.readthedocs.io/en/stable

[17] apispec, https://apispec.readthedocs.io/en/stable

[18] Swagger, https://swagger.io/specification/.

[19] JSX Specification, https://facebook.github.io/jsx/.

[20] M. Oskarsson *et al.*, "MXCuBE3 Bringing MX Experiments to the WEB", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, pp. 180-185. doi:10.18429/JACoW-ICALEPCS2017-TUBPL05

[21] immutable-js, https://github.com/immutable-js/immutable-js

[22] seamless-immutable, https://github.com/rtfeldman/seamless-immutable

[23] immer, https://github.com/immerjs/immer

[24] axios, https://github.com/axios/axios

[25] socket.io-client, https://github.com/socketio/socket.io-client

[26] A. Round *et al.*, "BioSAXS Sample Changer: a robotic sample changer for rapid and reliable high-throughput X-ray solution scattering experiments," *Acta Crystallogr. D Biol. Crystallogr.*, vol. 71, no. pt 1, pp. 67-75, Jan. 2015.