Improving FOLIO Architecture

Julian Ladisch
Verbundzentrale des GBV (VZG) in Göttingen, Germany

Martina Tumulla
North Rhine-Westphalian Library Service Centre (hbz) in Cologne, Germany

European Library Automation Group (ELAG) Conference
Berlin, May 8th, 2019
Overview and Project structure
FOLIO

• Aim
  − Development of an open source Library Service Platform (LSP)
  − Open, sustainable, innovative, flexible, expandable
  − Software for librarians to manage daily work

• Target Group
  − Academic and research libraries
FOLIO

• FOLIO is a product = software
• FOLIO is a community

• Development since 2016
  – Founded as open source project by the stakeholders EBSCO, Index Data and the OLE community (Open Library Environment)
  – Designed and developed cooperatively
Stakeholder – EBSCO

- Funding of contracted developer teams, e.g. 25 FTEs EPAM
- Bringing in own human resources: product management, product owners, developers, UX/UI designers
- Financial support of OLF infrastructure
- Funding of expert reports, e.g. technical evaluation (OTS-report), security audit (planned)
Stakeholder – Index Data

• Bringing in own developer teams, product owners and UX/UI designers
• Responsible for the basic technical architecture (Okapi)
• In addition: developers under contract with EBSCO
Stakeholder – OLE Community

• Funding by membership fees and Andrew W. Mellon Foundation
• Funding of developers and OLE staff
• Bringing in own human resources:
  Product owners, developers, functional experts, project management
• Financial support of OLF infrastructure
• Support of OLE partner projects, e.g. ERM apps
OLE Community

• OLE Board
• OLE Steering Committee
• OLE Managing Director
• OLE Director of Strategies
• OLE Project Manager
Open Library Foundation (OLF)

• New administrative home since 2016
  − Provides infrastructure (Confluence, Jira, Slack, …) and secures open source code (GitHub) for projects in higher education
  − Projects: FOLIO, OLE, GOKb, ReShare …
## FOLIO – Special Interest Groups

### SIGs und subgroups

<table>
<thead>
<tr>
<th>Metadata management</th>
<th>User management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data import / export</td>
<td>Internationalization</td>
</tr>
<tr>
<td>MARCcat</td>
<td>Consortia</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Resource access</th>
<th>Reporting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loans, reserves, requests</td>
<td>Reporting prototype subgroup</td>
</tr>
<tr>
<td>Printing slips, Patron notices, calendar</td>
<td>Privacy</td>
</tr>
<tr>
<td>Off-site integration</td>
<td>Accessibility</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Resource management</th>
<th>System operations and management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acquisitions small group</td>
<td>Data migration subgroup</td>
</tr>
<tr>
<td>App interaction group</td>
<td></td>
</tr>
<tr>
<td>ERM subgroup</td>
<td></td>
</tr>
</tbody>
</table>

| ERM subgroup         | |
| Privacy              | |
| Accessibility        | |
| System operations and management | |
| Data migration subgroup | |
FOLIO Releases – Q1 Bellis, April 2019

• Enhancement of basic functionalities
  – Metadata management
  – Circulation
  – Import, note fields, document storage
  – Acquisition: ordering, receiving/check-in
  – ERM: usage statistics (eUsage app)
  – ERM: license-, agreement- and package management
<table>
<thead>
<tr>
<th>Active</th>
<th>Name</th>
<th>Barcode</th>
<th>Patron group</th>
<th>User group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Bailey, Khalid</td>
<td>076102847180800</td>
<td>faculty</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Batz, Maddison Margie</td>
<td>144435285341195</td>
<td>staff</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Becker, Xavier Randal</td>
<td>44780165368253</td>
<td>graduate</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Bernhard, Sandra Name</td>
<td>246503218716161</td>
<td>undergrad</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Bradte, Jonathan Randi</td>
<td>78243336020693</td>
<td>graduate</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Christiansen, Tomas</td>
<td>8272159087407</td>
<td>staff</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Collier, Markus Paul</td>
<td>58015995900965</td>
<td>graduate</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Connelly, Audie</td>
<td>1631388355164</td>
<td>graduate</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Cummings, Chester Greg</td>
<td>32586189833208</td>
<td>graduate</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Dooley, Newton Scarlett</td>
<td>40370804414844</td>
<td>faculty</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Ebert, Shaylee</td>
<td>180816517350763</td>
<td>staff</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Feeney, Margret Tavanes</td>
<td>414876437022305</td>
<td>staff</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Fell, Aubrey Tia</td>
<td>267240097023773</td>
<td>staff</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Ferry, Bernadette</td>
<td>1766653708351</td>
<td>graduate</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Fritsch, Telly Lillie</td>
<td>3604183388288</td>
<td>graduate</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Gerhold, Sedrick Jayson</td>
<td>42051960062694</td>
<td>graduate</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Goczczyn, Anna Barnita</td>
<td>1844187064074</td>
<td>staff</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Gottlieb, Abbey Abbey</td>
<td>134294988317941</td>
<td>graduate</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Graham, Beaulah</td>
<td>265035487636620</td>
<td>graduate</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Hilpert, Arthur Kaci</td>
<td>746598380177</td>
<td>graduate</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Hirthe, Erna Trudie</td>
<td>655204529842836</td>
<td>faculty</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Jewess, Kalya</td>
<td>5283687895623</td>
<td>graduate</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Jewess, Karl Bernard</td>
<td>66209555976421</td>
<td>graduate</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Kessler, Cade Treva</td>
<td>831099341670498</td>
<td>faculty</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Klute-Folio, 2, Uschi</td>
<td>06180062573</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Kreiger, Barrett Devin</td>
<td>481573440960549</td>
<td>graduate</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Littel, Sofia</td>
<td>757563877102314</td>
<td>staff</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Maeyert, Lisette</td>
<td>66161721835573</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Champlin, Dagmar Emerald

**User information**
- **Record last updated:** 05/04/2019 11:18
- **Last name:** Champlin
- **First name:** Dagmar
- **Middle name:** Emerald
- **Barcode:** 781434363629411
- **Patron group:** staff
- **Status:** active
- **Expiration date:** 19/02/2020
FOLIO

Architecture
Technical Concept

• Open platform: Library Service Platform (LSP)
• Platform provides infrastructure for functional modules
• Functional modules → self-contained programs
  ➢ Can be developed independently from each other
  ➢ Can be selected and installed one-by-one
  ➢ Communication through interfaces
• Design based on micro-services idea
Technical Concept

- Promotes various support models
  - Cloud based, hosting, local
  - Commercial, library network, self
- Multi-tenancy
- Flexibly extendable, modular
- „Plug and play“ application
- Based on today’s requirements and aiming to future needs
Platform design

„APIs all the way down“

• This means that
  ➢ any developer can interact with any layer in the platform, and
  ➢ no component is too big to be replaced
Technologies

UI Toolkit, named Stripes
FOLIO comes with a default User Interface for the platform applications. At the same time, libraries or developers can take advantage of the UI toolkit to create a new UI as needed. The UI toolkit leverages the React framework, an open JavaScript library for creating user interfaces.

Basic FOLIO LMS Apps
Open source apps, e.g. ERM, acquisition, circulation, cataloguing, … New apps can be developed, free choice of programming language

Other Apps
Build on existing apps or develop new apps, commercially or open source

FOLIO Gateway
APIs, “Switchboard” (=communication) between user interface layer and the database. Tenant separation, communication via HTTP

System Layer
Central data layer (database, e.g. SQL) Indexing, Logging, tenant configuration
Technologies

Modern software stack of proven components

Frontend (= in the browser)

• JavaScript (ECMAScript 6)
• React/Redux

Backend (= on the server)

• Java 8 (Java 11 soon)
• Vert.x (asynchronous communication)
• RAML
• PostgreSQL
  ➢ JSONB (NoSQL) and
  ➢ Relational SQL
React and Redux

• Are open source JavaScript web frameworks for single-page applications (SPAs)

• React provides a framework for rendering user-interface components

• Redux is a data container that makes reading from and writing to the backend easy

• [https://reactjs.org/](https://reactjs.org/) und [https://redux.js.org/](https://redux.js.org/)
Stripes

- JavaScript program library for frontend modules
- Based on React + Redux
- Customized to Okapi’s and FOLIO’s needs
  - Communication via Okapi to backend modules
  - Granular user rights
  - Locale (language, date format, …)
  - Hotkeys (keyboard shortcuts)
  - Logging via Okapi

- [https://github.com/folio-org/stripes-core/#readme](https://github.com/folio-org/stripes-core/#readme)
vert.x

- Library for Java
- Facilitates simple concurrency
- Avoids many problems of parallel programming
- Asynchronous communication
  - Vert.x wraps a synchronous HTTP REST request into an asynchronous interface
- Reactive programming
- Design pattern “Reactor”

- [https://vertx.io/](https://vertx.io/)
RAML

• RAML = RESTful API Modeling Language
• Describes the interface of any module
• Generators take a RAML file and generate
  − Interface documentation: https://dev.folio.org/doc/api/
  − Java Code (Interfaces)
  − Validation, invoked by Okapi when calling an interface:
    • Sufficient user permissions?
    • Correct data format?
• https://github.com/folio-org/raml-module-builder
Database selection

• PostgreSQL
  – 2016 MongoDB proof of concept
• PostgreSQL became the DBMS of choice because it support both (!)
  – relational SQL database model
  – document based NoSQL database model
• NoSQL = Not-only-SQL, in this case document based (JSON documents)
• PostgreSQL can process JSON documents as JSONB, this is an efficient binary format where the JSON document is decomposed allowing indexing
JSON

• JSON = JavaScript Object Notation
• FOLIO stores most of the data as JSONB
• Data exchange format of most FOLIO APIs is JSON
• Vert.x offers extensive JSON support, JSON is vert.x’ main exchange format
• JSON is a very common data exchange format for asynchronous browser server communication
  – This applies to Java as well
Database operation

• Each storage module may start an own PostgreSQL instance
  – We use this for software development
• Use a parameter to connect an external PostgreSQL installation
  – We use this for our demo and test installations
  – Allows for high availability and replication with PostgreSQL cluster
Inter-module Communication

Example: Check-out app uses three backend modules – the module mod-circulation combines the loan data received from mod-circulation-storage with the title data from mod-inventory-storage and returns the merged dataset.

*-storage = database abstraction layer
Okapi

• Okapi implements API gateway pattern
• Is the tenant allowed to access the module?
• If several versions for one module run:
  − Select the version that has been activated for the tenant
• Has the user sufficient access rights?
• Validating the parameters passed into the API
• Passing the API request to the module
• https://github.com/folio-org/okapi
Modules

• Modules communicate via interfaces *only*
• Independence
• Easy to maintain, easy to exchange
• License can be selected independently per module:
  – Proprietary
  – Viral license like GPL or AGPL
  – Permissive free license like Apache or MIT
• Select programming language and software libraries independently
  – Core modules use the same software stack (Java, Vert.x, …)
App and Module Architecture

• Functionality is split into apps by business area
• This reduces inter-app data exchange
• Example: Check-out and Check-in are one app only
  – Even if there are two buttons on the user interface
• Not: Nanoservices with mini modules
• One developer team per app
• An app usually has a GUI module, a business login module and a data storage (access to database) module
Vagrant und Docker

• Install a complete FOLIO system using Ansible:
  – https://github.com/folio-org/folio-ansible

• Manual installation with explanations:

• All modules as Docker containers:
  – https://hub.docker.com/u/folioorg/
  – https://hub.docker.com/u/folioci/

• Download a complete FOLIO system as a Vagrant box:
  – https://app.vagrantup.com/folio
FOLIO

Technical Evaluation
Technical Evaluation

• The technical basis and the architecture of the FOLIO platform have been evaluated for three times:
  - by members of the OLE-Community
  - by EBSCO
  - by Open Tech Strategies (OTS)

• All evaluations were successful, and the suggestions resulted in improvements or prioritized issues

• The OTS report from January 2019 is online
FOLIO

Accessibility
Accessibility in FOLIO

Accessibility = easy to use for all, including those with disabilities
FOLIO aims at WCAG 2.1 priority AA compliance

WCAG:
• Web Content Accessibility Guidelines
• International standard for accessibility
• Required by law in many countries
• European Union law requires new websites of public sector bodies to comply with WCAG 2.1 priority AA from 23 September 2019 on
Stripes Accessibility

- Stripes is FOLIO‘s GUI toolkit, provides reusable components
- Designed to be accessible
- Accessibility and usability is checked on a regular basis
  - In usability labs
  - During our monthly power hour
- Feedback improved Stripes components and the guidelines
- Accessibility architecture = built-in by design

https://wiki.folio.org/display/A11Y
https://ux.folio.org/docs/guidelines/accessibility/
FOLIO

Query language –
From CQL to GraphQL
Query language – from CQL to GraphQL

• CQL = Contextual Query Language
  − is a DBMS agnostic query language
  − is used by the front-end and by back-end modules that query data records from other back-end modules

• CQL has limitations

• Solution: A GraphQL module was added

• GraphQL supports advanced and complex queries

• This architectural improvement was possible because of FOLIO’s microservices-like architecture

• [https://dev.folio.org/reference/glossary/#cql](https://dev.folio.org/reference/glossary/#cql)
• [https://github.com/folio-org/mod-graphql](https://github.com/folio-org/mod-graphql)
Example GraphQL query

```graphql
query {
  instance_storage_instances(query: "title=baby") {
    totalRecords
    instances {
      title
      holdingsRecords2 {
        callNumber
        holdingsItems {
          barcode
        }
      }
    }
  }
}
```

- Join three tables: instance, holdings, item
- Return selected fields only: totalRecords, title, callNumbers, barcode
Tenant in FOLIO

- Tenant = completely independent institution
  - Branch library is not a tenant and uses granular hierarchical access rights.
- FOLIO supports cloud installations
- Tenants share cloud hardware and cloud software
- Strict tenant separation required
- For each combination of tenant and module we create a database user and a logical database:
  
  ```
  CREATE ROLE ${university}_${module};
  CREATE SCHEMA ${university}_${module} AUTHORIZATION ${university}_${module};
  ```
- We have a tenantSeparation unit test
Tenant in FOLIO

• Each module runs the CREATE ROLE and CREATE SCHEMA commands when a new tenant needs to be activated

• Okapi passes the credentials of a database superuser to the module

• This is an architectural deficiency for security reasons

• Better design:
  – Only a central service has superuser rights
  – and creates the role and the schema
  – and passes the information to the module

• Architectural change recommended by OTS report and on the way
FOLIO

Additional DBMS support
DBMS support: PostgreSQL + ?

• Any back-end module may use any DBMS
• RAML Module Builder (RMB) is a FOLIO software library
  – Supports only PostgreSQL
  – Supports PostgreSQL JSONB columns
  – Reduces boilerplate code for each module
• Using RMB is the most easy way
• Most modules use it
• Some use Grails with PostgreSQL instead
  – mod-licenses, mod-agreements
DBMS support: PostgreSQL + ?

- OTS report recommends additional DBMS back-ends
- FOLIO has postponed decision

- What do you think?
- How should FOLIO prioritize it?
Speakers

Martina Tumulla

works as a systems librarian at the North Rhine-Westphalian Library Service Centre (hbz) in Cologne, Germany. She supports FOLIO’s development as co-convener of ERM subgroup and is member of Resource Management SIG, Consortia SIG and Product Council.

tumulla@hbz-nrw.de

Julian Ladisch

works as a senior developer at the headquarters of GBV in Göttingen, Germany, and is active in the FOLIO project since its beginning in 2016. He is a member of the FOLIO platform core developer team.

julian.ladisch@gbv.de
FOLIO Links

- Code on GitHub
  https://github.com/folio-org
- Dev Wiki
  https://dev.folio.org/
- Demo Installation
  https://folio-demo.gbv.de/
  https://folio-demo.hbz-nrw.de/
  (diku_admin / admin)
- FOLIO project website (in German)
  https://www.folio-bib.org/
- FOLIO Wiki
  https://wiki.folio.org/
- OLE Community
  https://www.openlibraryenvironment.org/
- Open Library Foundation (OLF)
  http://www.openlibraryfoundation.org/
- FOLIO
  https://www.folio.org/
Attribution-NonCommercial 4.0 International

The text of this presentation is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License: https://creativecommons.org/licenses/by-nc/4.0/