Architectural Beeprints

WOLFcon 2023

Chicago, USA

Tod Olson & Vince Bareau

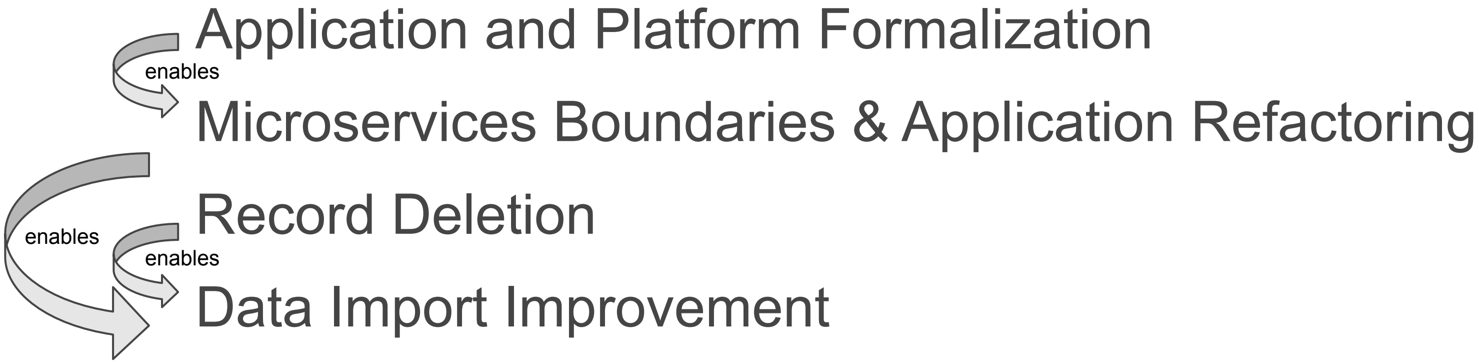
Overview

FOLIO has reached a certain level of maturity, and is in production at a significant number of libraries.

FOLIO is at a plateau but needs to evolve further to support increasing needs of scale and complexity.

This forum will look at some aspects of FOLIO that need to mature, building on ideas presented last year and on recent work.

Agenda



Application and Platform Formalization

enables

Microservices Boundaries & Application Refactoring

enables

Record Deletion

enables

Data Import Improvement

Linked Data and Inventory

Application and Platform Formalization

Application and Platform Formalization

At WOLFcon22 several enabler concepts were introduced, including that of Application Formalization

In the year since, there has been progress.

- Many discussions were had
- The problem space was expanded beyond Applications to also include Platforms
- **Designs have begun to materialize**

Application and Platform Formalization

Definitions for Applications and Platforms have been proposed:

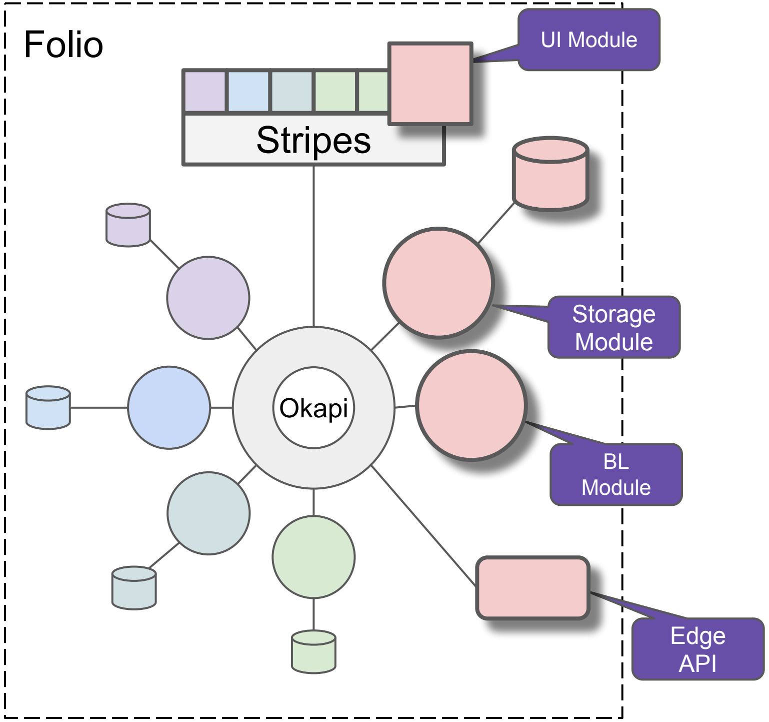
A Folio Application is the minimal but complete set of elements which together are intended to deliver a specific solution to Folio

A Folio Platform is a particular combination of Folio Applications together with the Folio Core which delivers a specific system solution

Application and Platform Formalization

Applications are formalized by an **Application Descriptor**

- References the necessary Module Descriptors
- Vertical alignment of related Front-end and Back-end modules

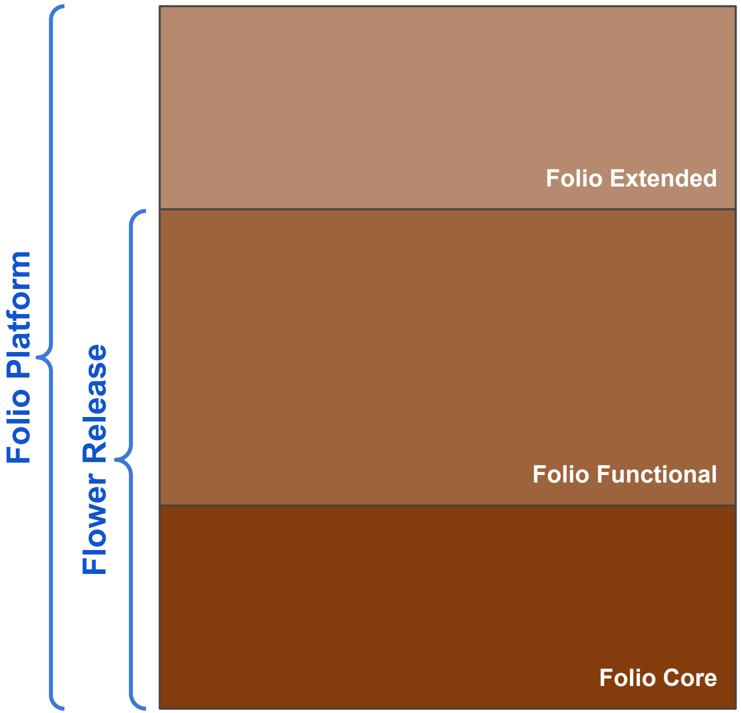


Application and Platform Formalization

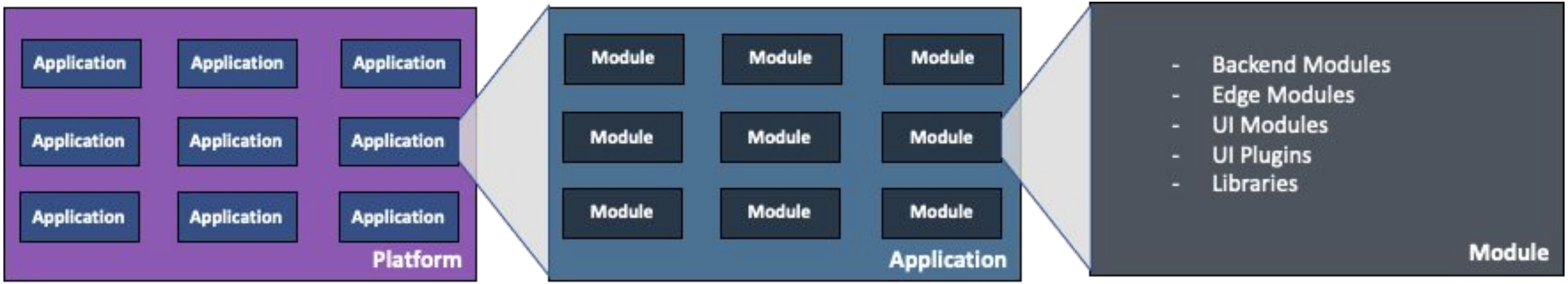
Platforms are many and tiered

Platforms are formalized by a **Platform Descriptor**:

- References the necessary Application Descriptors



Platform - Application - Module : A Hierarchy



Platforms:
- contain Applications
- provide a **Platform Descriptor**
- have their own versioning
- There is a special type of Platform: Core

Applications:
- contain Modules
- provide an **Application Descriptor**
- have their own versioning

Modules:
- The existing concept of Folio modules and components (all types)
- provide a **Module Descriptor**
- have their own versioning

Application and Platform Formalization

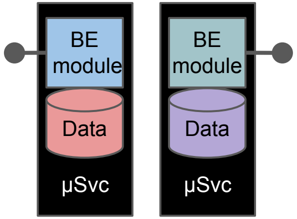
Formalizing Applications and Platforms is an enabler to
Microservices Boundaries & Application Refactoring

Microservice Boundaries
&
Application Refactoring

Enabler: Microservice Boundaries (WOLFcon22)

Where we are

- Folio embraces microservices (sorta)
- Each backend module is considered a µservice.
- Each µservice is responsible for their own data layer.
- Only the backend module responsible for data storage is allowed to directly access it
- All other data access must be through the µservice API interfaces.
- Some older applications have implemented a separation between “business logic” and “storage” modules.

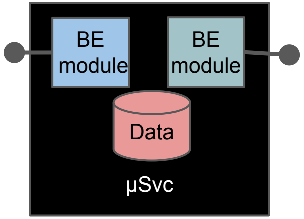


Ideas for Discussion

- Folio continues to embrace µservices (more so)
- The **Bounded Context** is considered a µservice
- A Bounded Context may consist of several backend modules
- All modules in the Bounded Context may directly access the shared data storage
- The “storage” vs. “business logic” pattern is eliminated

Benefits

- **Dependency Reduction**: reduces the number of modules
- **Ease of Development**
- Reduces latency (reduces http traffic)
- Contributes to **Data Integrity**
- Allows for extension modules



Application Refactoring/Microservice Boundaries

Background

- FOLIO has grown considerably
- Current µservice boundaries remain at the module level
- Number of moules has increased, resulting in even more API-level dependencies
- Increased complexity in cross-app interactions
- Thinking on Application Formalization has matured (an enabler)

Pressures

- More deployments, dev & operations overhead
- Need for these benefits is more urgent

Opportunities

- Rethink storage layout for the µservice (cf. comments on SRS and **Data Import**)
- Simplify cross-app data flow
- Rethink the single source of truth:
  - µservice truth vs centralized truth
- Transactional integrity at the µservice level
- Go beyond mere CRUD APIs: actions vs records
- Streamline communications between boundaries: cross-app or cross-tenant

Application Refactoring/Microservice Boundaries

Microservices Boundaries/Application Refactoring is enabled by *Formalizing Applications and Platforms*.

But it is also an enabler for *Data Import Improvement*

Record Deletion

Record Deletion Functionality

- Folio is lacking the ability to delete records
- Many workflows cause orphaned records
- Accumulation of orphaned records slows down Folio as a whole

Deletion of records is not a trivial task

- Involves identifying all possible dependencies on a particular resource
- Requires a rules-based workflow to clear a record for deletion.
- Records must be soft-deleted in case of mistakes
- Rules must cross Folio application boundaries
- Now Rules must even cross tenant boundaries

Record Deletion Functionality

A design has been proposed

- Soft delete
  - Archive deleted records
  - Supporting rollback
  - Requires a companion hard delete functionality
    - Customizable at the customer-level
- Proposed pilot for Inventory items/holdings/instances
  - Waiting for capacity planning SIG
- Rule-based dependency checking
- Crosses application boundary!

Record Deletion Functionality

Record Deletion is an enabler for further Data Import functionality.

Data Import Improvements

Data Import Improvements

Important improvements have been made in recent months to Data Import.

However, even more improvements are possible, if not necessary, to address the scalability of Data Import

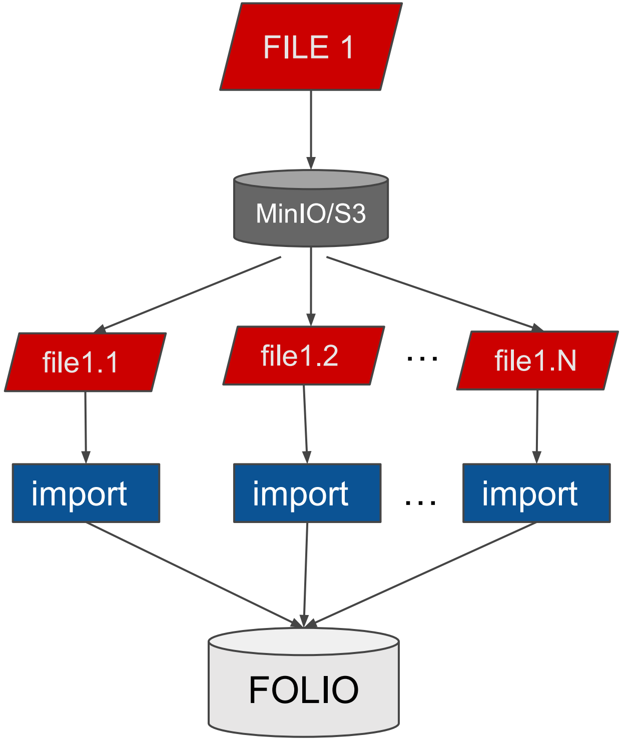
Data Import Improvements - Upload Process

File Chunking

- Support upload to MinIO/S3 storage
  - instead of current local storage on EC2
  - applies to ALL uploads even those going through the UI
- Automate slicing files into smaller (optimal) parts
  - Happens before import
- Submit as child jobs under an "umbrella" parent job
- UI changes to reflect parent job vs child jobs

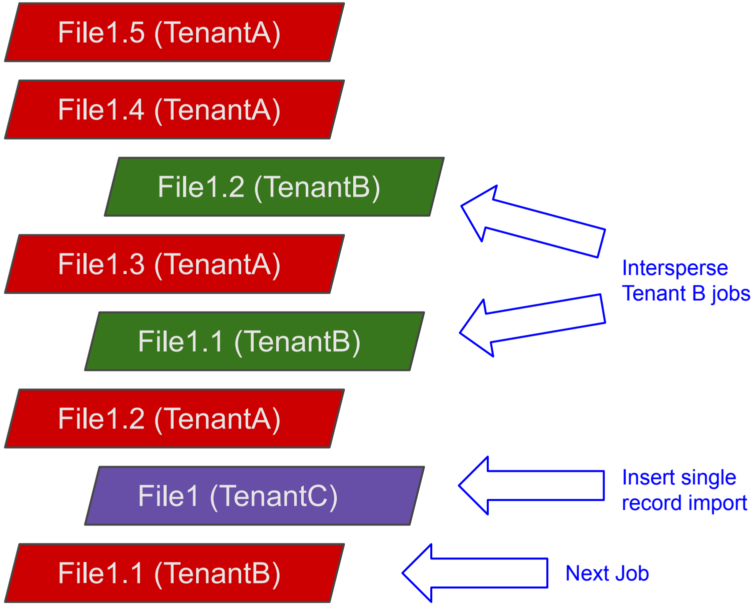
Parallel Processing

- Because all jobs go through S3 instead of local storage, several instances of DI can be run in parallel



Data Import Improvements - Queue Management

- Store info about jobs in DB
  - Cross-tenant storage which reflects ALL import activity from ALL tenants
- Prioritization
  - Take into account size and which tenant a job belongs to when selecting next job to process
    - Pickup smallest jobs first and also consider the total number of jobs for a given tenant
    - Check running jobs -> if one tenant has all jobs running then pick next job from another
  - Algorithm is optimized differently for single-tenant vs multi-tenant deployment
- Benefits Single record import
  - Insert between fragments of large jobs (e.g. quickmarc) - existing functionality



Data Import Improvements - Future

Potential Database changes

- Relational DB model for SRS records instead of current JSONB
  - JSONB currently used in storage: expensive in CPU and parsing
  - Bottleneck to DI
- CURRENTLY: Side-by-side testing of relational vs JSONB
  - Use simple relational table to store content of MARC records => SRS
  - Measure SQL queries with JSONB vs same queries in relational tables via API calls. Same datasets in each case
- FUTURE: if successful think about change for mod-inventory too

Other Potential Improvements

- Make SRS storage relational
- Make Inventory storage relational
- Add bulk APIs and operations
- Garbage collection in SRS
  - **Record Deletion**
  - Errors will leave MARC SRS records orphaned.
  - Find and delete detached MARC records.
- Create better logging and diagnostics.
- Additional prioritization logic for queue management.
  - Maybe slice very small child jobs so less waiting for other jobs to be picked up
- Integrate with **Linked Data and BIBFRAME** pipeline from LoC

Linked Data and Inventory

Linked Data and Inventory

Background

- Since the very start, Folio has sought to embrace BIBFRAME2 as a data model
- Though not completely tied to MARC, MARC is currently the only format for which support is implemented in Folio
- Using the existing data models, each new format adoption will cause significant (linear) expansion of large parts of Folio (Inventory, SRS, Data Import, Data Export, Bulk Edit etc…)

Pressures:

- The Library of Congress is adopting Folio and promoting BIBFRAME and Linked Data
- **Data Import** needs to support more formats

Opportunities

- Folio to embrace Linked Data and BIBRAME
- Refactor Inventory
  - Adopt data graph
  - Simplify support for multiple metadata formats
- Adopt BIBFRAME editor(s)

Wrap up
and
Thank you

Appendix

WOLFcon 2022 Presentation

https://docs.google.com/presentation/d/15_t3W-AQ2SA-kYuLTclYUdjR18wY-uJ8lpcT1Rxehg8/edit#slide=id.p