Architecture Blueprint Sessions - WolfCon 2020

Two sessions were used during WolfCon 2020 to discuss topics that relate to the Folio Architectural Blueprint. The Architecture Blueprint being defined as those longer term platform capabilities which are either strategic or foundational for future feature development. The outlook is forward looking, as compared to technical debt which looks to the past.

The format of these sessions was to allocate 10 minutes for each topic. After a brief introduction each topic was discussed with in the room. The goals of these sessions were:

- inform on items being considered for a formal architectural blueprint
- gather feedback on those items
- obtain a sense of the perceived urgency around each
- NOT to delve into the solution space for each item

Blueprint Items Discussed

Day 1:

- Security (fallout of the Security Audit)
- Refactoring Okapi
- Tenant Management
- Multi-Tenancy and Cross-Tenancy
- Adopting PubSub
- Adding Support for GDPR

Day 2:

- Search Engine
- Users and Permissions
- Automation Engine
- GraphQL
- Database Connectivity

Not Discussed:

- Codex
- Inter-Folio Integration

Materials

Folio Architectu...WolfCon2020).pdf

Recordings of Sessions

Day 1:https://drive.google.com/open?id=15mAQu3u0CEbR1AX2NrjfL0A6mHHFuvM9

Day 2: https://drive.google.com/open?id=1ZR2PRTKHro9HRaNxzQHbDRGZBNu-OBhVArch

Session Notes

Thanks to Zak

Day 1:

```
many topics
forward looking vs backward looking (tech debt)

security audit fallout
    audit in February; expect results ~march 1
    weigh the urgency of the fallout vs other changes?
    there may be changes that roll together with other non-security changes
    urgency is high: expect to act on it basically as soon at it comes out
    we have a security policy but not yet a team; whoops; we should do that ASAP
        doc exists but not yet public
        will be publicized when ready
            security@folio.org, non-public jira project
        job of sec team to triage those jobs
            determine whom to notify, and when, and with how much detail
    emergency release process?
        not in place at present.
        hosting provider may provide a work-around to limit the impact
        at the same time, issue will be raised with the community to dev a fix
        once impl'ed, verified, then SP can deploy to hosting instance
        permanent fix may take a while...
        how does this interact with q'ly release schedule?
    general agreement: yes, respond to the security audit

refactoring okapi
    okapi has grown in role since original conception, impl
    many features
        proxy gateway
        tenant
        dep mgmt (build time)
        discovery system (installed modulues)
        tenant APIs (provisioning, upgrades)
        etc etc: timer, pre/post filters, etc
    goal: separate into multiple components
        high risk of making change to a monolith
        is security vulnerability
            e.g. setup requires elevated perms
            at the same time, stripes talks to okapi
            this is not ideal; these two are incompatible
        goal: restricted perms on runtime role
            elevated perms on setup role
        separate tools are more free to evolve because are independent
    splitting also allows these components to scale independently
    dep mgmt: does this compound that problem? still more apps, ugh
        (devil's advocate)
        these aren't apps; are tools.
        yes, proliferation is a problem, but separation of concerns is worthwhile
        can separate services without necessarily separating codebases
            this would be nice; could resolve some issues of compatibility
    don't have to separate each services into separate modules
        but some separation feels like a good thing
    general agreement to discuss this further, esp WRT security
        but uncertainty about the details of this refactoring
    other lurking issues: performance, efficiency
    what of shanghai's reqs: okapi must change to accomodate that, or wholesale changes to comm model
        separating discovery from gateway, don't have to change gateway
    maybe refactor is a misnomer here; we are discussing outward facing changes
        ... not simply internal refactoring
    a discussion about okapi is a platform discussion
        this is re-architecture, not refactor
        (this is scarier; has bigger impact on the rest of the platform)
        can change process model without having to change everything
        refactor now to make re-arch later less difficult?
    agree this is blueprint item
        timing may depend on sec audit
        discuss this in parallel/immediately after the sec audit?
    agree to discuss soon-ish

tenant management
    currently owned by okapi, at least in part
    1. admin component for provisioning, upgrading tenants
    2. runtime component for tenant registration
    WRT updates/migrations:
        part of modules?
        part of tenant API?
        devops sees different perspective
    may not matter where the tenant API impl lives, i.e. in okapi or elsewhere
    migrations need the elevated perms; can run on BE as admin tool if separated
    security aspect is the silver lining of this: isolating this from Okapi is good
    this is def. part of okapi refactoring
    v. impt for multi-tenancy arch.
    present issue is highly isolated tenant fx'ality; to coordinate across tenants, must build it separately
    does this separation make simple cases simple if you are not impl'ing multi-tenant?
        WRT tenant mgmt only, for single tenant is minor help
        for multi tenant, is HUGE help
    this also affects multiple tenants using one okapi
    this is dependent on multi-tenant libs planning to go-live in summer 2020
    agree this is a blueprint item
    important, but less so than security
    what is the price of this kind of change after a multi-tenant place goes live?
        depends on the change....
        maybe is huge: can't go live until have multi-tenancy, and need this first....
    maybe must include proxy as part of this?
    this also addresses the consortia problem
    investigate how reshare handles this?
        they run fully isolated tenants, exchange data across tenants
    need clear understanding of motivation for these items
    what is full multi-tenancy?
        multiple tenants sharing data; this does not exist at present
    agree to discuss ... "slater"
    does this even belong in okapi if it stays within Okapi?
        don't necessarily have to break into own service

full multi-tenancy
    how do we provide full multi-tenancy
    MT means sharing data across tenants
    want to separate initiating tenant from target tenant of an action
    this is maybe more comfortable as a roadmap item than tenant-mgmt
    must talk about this: may or may not imply LOTS of work
    there's a lot of complexity here
    permissions are currently tenant-scoped; need a new perm system
    talk about this sooner in order to prioritize it sooner
    security is an impt consideration here: isolation provides security; people want that
    the soln here must not undermine isolation elsewhere
    agree this is blueprint item
    discuss sooner

adopt pubsub
    have dev'ed P/S as part of source-record-storage
    so can handle comm between modules with event-based mechanism
        developed with this in mind. yay!
    decouples modules from one another
    overlaps with techdebt somewhat
        must reexamine some impls;
        can also
    discovery: realtime upgrades is holy grail
        so from vufind's view: this is HUGE
    very strong feelings about this being essential to future extensibility
        folio-core is hard to break apart right now
        having this is more of a hook module
        can have more uni-directional deps
        strong agreement all around
    do we impl/reimpl some integrations?
    most urgent is to make sure it is present somewhere?
        not part of edelweiss, is on master now?
    folijet to demo
    this is the beginning of Saga support
    could provide for distributed transactions in some circ functions
    strongly agree this is a blueprint item
    discuss soon, assuming is actually available

support for GDPR
    support is no brainer; must do
    mostly concern for hosting provider
    right of access, rectification, erasure, restrict processing, data portability
    can do some of this now but is highly manual process
    was always intended to make this easier by providing some APIs
    arch req not just a feature req because imposes some req's on modules
    does GDPR give you a timeline? believe yes but don't know
    this is crucial for this to be a non-US centric project
        not urgent, later is OK but is required
    Chalmers is live; this puts pressure on them.
        manual process is OK now, but won't be at scale
```

Day 2:

```
goal
    id issues NOT solve them
    then prioritize, set a timeline

search engine
    search built on top of postgres at present
    but will hit a tipping point where that starts to fail
    next level: dedicated search engine, e.g. elasticsearch
    is it functional, performant, easy to implement
    what about results display?
    universal search is a separate/related question
    is peristence engine not best for postgres? e.g. cassandra?
    this proposal suggests search as a bolt-on
        changing the persistence mechanism would be more integrated
    full-text engine vs rdbms have different strengths; may need both
    OLE used solr: problem then was then that solr indices became treated as source of truth
    postgresql is a swiss army knife of persistence
        rdbms, message queue, json storage...
    discussions WRT postgres services may make HA instances hard
        but adding other services, then have to solve those same problems for add'l tools
    any solution will take massaging to get good perf
    fulltext search is powerful ... but doesn't support localization
        will have to wrestle with this eventually
    NLAustralia: we have gone down the search engine road; happy with it
        do we have exp with large DBs?
        we know the lucene-based search indexes that perform fine
        if we lack the experience, we shouldn't research it, we should find ppl with exp
            this knowledge exists; don't spend time rediscovering it
    discovery layer will surely exist separately, and surely use a search engine
        but we are building an ILS, not a discover layer
    agreement this is impt
    agreement on sooner rather than later

users and permissions
    introduce tenant-level and system-level users
    every user in the system now gets perms assigned
    introduce role-based perms
    at system level, have batch operations
        don't want to create dedicated users for this, is hacky
    1. system-level users
    2. roles
        abstraction layer between users and roles
        will be impt when workflows; e.g. notifications assigned to roles
    can assign multiple psets to a single user
        psets aggregate permissions
        roles would aggregate users
    no way to cluster users at present
    UChicago: this would be helpful
    roles provide hierarchical permission assignment
    teams: ERM has some notion of this
        folio doesn't have any data-level protections
        perms are focused around endpoints only
        could roles help get us there?
        e.g. a group of people who should only be able to see orders with a certain attr
            e.g. only math-dept orders, only history-dept orders...
    agreement: should we split tenant level/system level users
    agreement: should explore roles
    this will only get harder if we wait
    agreement: system user notion is more impt than roles

automation engine / "workflow engine"
    engine required to provide automation on the backend; NOT user-visible
    should/when will we adopt such an engine
    could eventually help with user-visible things, e.g. tasklists
    nuance: site specific tweaks to processes are necessary
        right: we are talking about building the player piano; of course scroll is changeable
    are there concerns that necessary hooks in the system may be absent?
    camunda POC attached to GitHub:folio-org, but lots of progress in GitHub:tamu
        TAMU: using workflow for data migration
    this would be unique in this marketplace! whoop whoop!
    any relation to pubsub? ATM, handled separately
    agreement: this belongs on roadmap
    agreement: sooner; there is Q3/Q4 work scheduled that is related
    ought to be able to do anything manually that is being automated

graphQL
    prototype exists
    allows you to tailor a single request to get data in desired shape
    behind that is system to traverse the graph to assemble this
    initial look: optimization for UI; fewer requests
    subsequent look: want to streamline dataflow between modules, not just in UI
        right now can't get only IDs; get full records, which is less efficient
    could streamline APIs?
    at simple impl level, could simply add biz layer atop APIs
        still making API calls under the hood though...
        don't know about opportunity for actual optimization
    graphql could maybe provide some efficiencies

database connectivity
    we have module level connections currently
        if have multi-tenant system, any tenant uses single cx
    would be great to have more granular connectivity
        e.g. assign per-tenant cx
        e.g. interface-level cx, e.g. some data in local stores, some in cloud...
        e.g. GET/PUT/POST separation, separate cx to read-optimized/write-optimized stores
    for folks wanting to share infrastructure with separate data, this does not work well
    last bullet is solved outside of folio; we should not solve it ourselves
    secret mgmt is related, maybe?
    agreement this is blueprint item
    agreement this is urgent

codex!
inter-folio integration
    CALIS has explored this somewhat
    how do multiple folios interact? transparently to user.
    agreement this is later
```